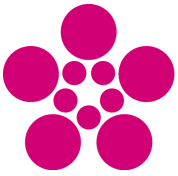


Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Ekonomická fakulta

Katedra matematiky a informatiky

Bakalářská práce

Implementace vysoce dostupného clusteru

Vypracoval: Zdeněk Heřman

Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2016

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Fakulta ekonomická
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Zdeněk HEŘMAN**
Osobní číslo:	**E13545**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Název tématu:	**Implementace počítačového clusteru s vysokou dostupností**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování:

Cílem bakalářské práce je navrhnout a popsat implementaci zvoleného počítačového clusteru s vysokou dostupností, který bude odolný proti výpadkům hardware.

Metodický postup:

1. Studium odborné literatury.
2. Obecný popis počítačového clusteru.
3. Teoretický popis konkrétních dostupných systémů.
4. Popis implementace zvoleného řešení.
5. Závěr.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **40 - 50 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **CISCO SYSTEMS, Inc.** *Data Center High Availability Clusters: Design Guide* **[online]. San Jose, CA: Cisco Systems, 2006. Text Part Number: OL-12518-01. Dostupné z: http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/HA_Clusters/HA_Clusters.pdf**
2. **LUCKE, Robert W.** *Building clustered Linux systems.* **Upper Saddle River, NJ: Prentice Hall PTR, c2005, xxxviii, 606 p. ISBN 978-0-1314-4853-7.**
3. **M.C. CHENG, Simon.** *Proxmox High Availability.* **Birmingham, UK: Packt, October 2014, 258 s. ISBN 978-1-78398-088-8.**
4. **PIEDAD, Floyd a Michael HAWKINS.** *High availability: design, techniques, and processes.* **Upper Saddle River, N.J.: Prentice Hall PTR, c2001, xvii, 266 p. ISBN 978-0-1309-6288-1.**
5. **VAN SANDER, Vugt.** *Pro Linux High Availability Clustering.* **1 edition. New York: Apress, July 28, 2014, 168 s. ISBN 978-1-4842-0080-3.**

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **9. ledna 2015**

Termín odevzdání bakalářské práce: **15. dubna 2016**

doc. Ing. Ladislav Rolínek, Ph.D.
děkan

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

L.S.

prof. RNDr. Pavel Tlustý, CSc.
vedoucí katedry

V Českých Budějovicích dne 27. března 2015

Prohlášení

Prohlašuji, že svoji bakalářskou/diplomovou práci jsem vypracoval/a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47 zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské/diplomové práce, a to – v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum 2.3.2016 Podpis studenta

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce, panu Mgr. Radimu Remešovi, za ochotu a cenné rady poskytované na konzultacích. Velké poděkování také patří celé mé rodině, hlavně manželce Zuzaně a dceři Verunce, za silnou podporu při psaní této práce.

Obsah

1. Úvod

1.1 Cíl práce

Primárním cílem je realizace počítačového clusteru s vysokou dostupností, který je odolný proti výpadkům hardware, počítačové sítě a případně i problémům s aktualizacemi software na serverech. Cluster bude sloužit jako hostitel pro virtuální servery poskytující důležité služby uživatelům. Zde bude kladen důraz na vysokou dostupnost těchto služeb.

Postupně bude docházet k seznámení, v čem je řešení s vysokou dostupností odlišné od běžného, jaké jsou důležité prvky tohoto systému, a co nám tato implementace počítačového clusteru s vysokou dostupností nabízí. Budou zmíněny alternativy, jak takového řešení dosáhnout. Nakonec bude uvedena ukázka realizace pomocí operačního systému Debian Linux s nadstavbou Proxmox.

2. Metodika

2.1 Teoretická část

Cluster s vysokou dostupností (zkráceně *HA cluster* z anglického označení high-availability cluster) je jedním ze základních systémů pro prostředí, kde je důležitá maximální dostupnost služeb a ochrana proti nečekaným událostem, jako je výpadek napájení, síťového přepínače, serveru, diskového pole atd. Dokonce v případě výpadku více komponent systému je možné i nadále poskytovat potřebné služby klientům, aniž by zaznamenali nějaký problém. Patří mezi základní stavební prvky informačních technologií velkých korporací (například Google, Microsoft, Seznam), bank nebo státních institucí. Použitím takového řešení lze dosáhnout téměř 100% dostupnost poskytovaných služeb.

Aby bylo možné takového stavu dosáhnout, je zapotřebí připravit dlouhodobou firemní strategii v oblasti informačních technologií. Podmínkou implementace HA clusteru je plná redundance použitých komponent, redundance dat, plně spolehlivý kanál pro komunikaci mezi jednotlivými uzly clusteru a operační systém, který HA clustering podporuje. Redundancí je v tomto případě myšleno zdvojení nebo vícenásobení jednotlivých částí systému. Redundantní musí být veškeré prvky, protože jinak v clusteru vzniká slabé místo, které se může stát zdrojem fatálních problémů.

Ve třetí kapitole je podrobný přehled a popis kritických prvků systému a je zde zmíněno jaké technologie lze použít.

2.2 Praktická část

Čtvrtá část této práce bude podrobným návodem, jak vysoce dostupný cluster implementovat. Celá softwarová část bude řešena pomocí opensource produktů. Obecným trendem v IT je virtualizace, a proto je zaměřena na cluster poskytující virtuální prostředí pro virtuální servery, které následně koncovým uživatelům nabízí počítačové služby. Operačním systémem pro jednotlivé uzly bude OS Linux z distribuce Debian ve verzi 7. Jako jádro celého clusteru bude použito rozšíření Proxmox. Krok za krokem je popsán postup, jak nainstalovat operační systém, jak jej nakonfigurovat a jak instalovat a nastavit rozšíření, které realizuje samotný cluster.

Závěrem budou zmíněny jaké výhody a nevýhody takové řešení přináší.

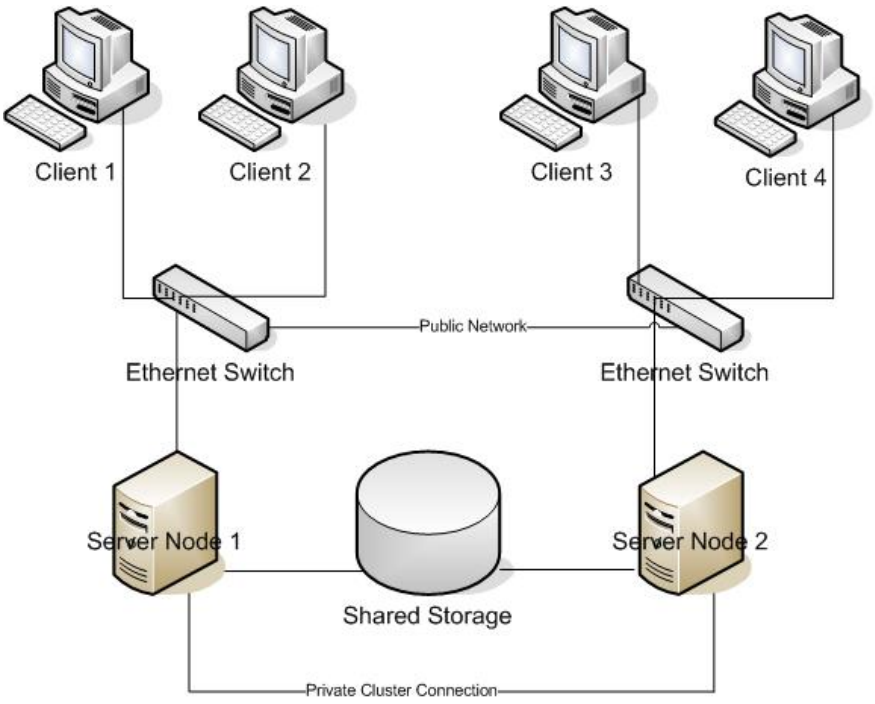
3. Přehled řešené problematiky

Tato kapitola je zaměřena na jednotlivé komponenty HA clusteru. Jsou vysvětleny některé základní pojmy. Záměrně jsou opomíjeny IT komponenty, které nejsou v žádném přímém vztahu s implementací. Jedná se o jakýsi seznam kritických částí clusteru a popis možných technologií, které je možno využít.

3.1 Počítačový cluster

Počítačový cluster je skupina počítačů, které jsou mezi sebou navzájem propojeny a spolupracují spolu. Každý počítač v této skupině se označuje **uzel** nebo anglickým slovem **node**. Uživatelům se takový systém může jevit jako jeden počítač.

Důvodem vytvoření clusteru může být zvýšení rychlosti nebo spolehlivosti, případně rozložení zátěže. Díky tomu je možné dosahovat vyššího výkonu nebo dostupnosti, než by bylo možné u jednoho počítače. Obrázek 3.1 ukazuje základní zapojení a prvky clusteru.

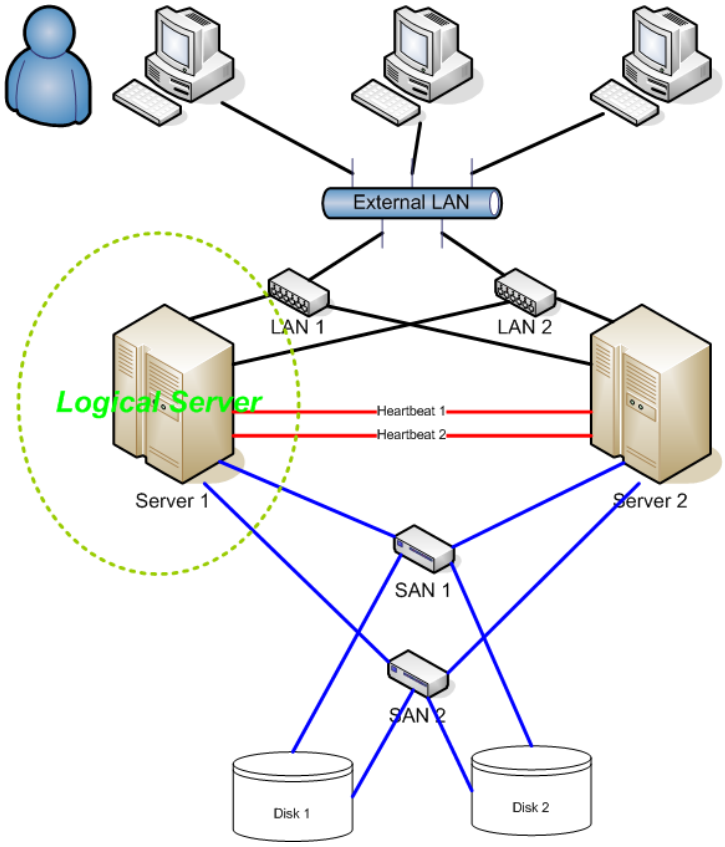


Obrázek 3.1: Schéma clusteru (Leicht, 2007)

3.2 Cluster s vysokou dostupností

Je druh clusteru, který je zaměřen na maximální spolehlivost poskytovaných služeb. Je navržen tak, aby bylo dosaženo co nejmenší pravděpodobnosti výpadku z důvodu chyby hardware, software, napájení, síťové konektivity nebo nutné údržby. Veškeré jeho části jsou plně **redundantní**, tedy každá část systému je minimálně jednou zálohovaná (obrázek 3.2). Systém dokáže, v případě výpadku nějaké z částí, automaticky přepnout na alternativní záložní řešení a to vše v řádu sekund. Tento proces automatického přepnutí bez nutnosti manuálního zásahu administrátora se nazývá **failover**. Díky tak rychlé reakci uživatelé nezpozorují žádný výpadek, případně chvilkovou pomalejší odezvu nebo velice krátkou nedostupnost systému.

Zajišťování vysoké dostupnosti může probíhat buď přímo na úrovni služeb, nebo na úrovni poskytování virtualizačního prostředí. V prvním případě cluster zajišťuje, že na daných portech alespoň na jednom z uzlů běží požadovaná služba. U virtualizačního prostředí cluster zajišťuje, že v rámci clusteru je na jednom z uzlů v provozu chráněný virtuální stroj. Pro toto prostředí je na uzly kladen výrazně vyšší požadavek na systémové zdroje, jako je dostatek procesorového výkonu a operační paměti.

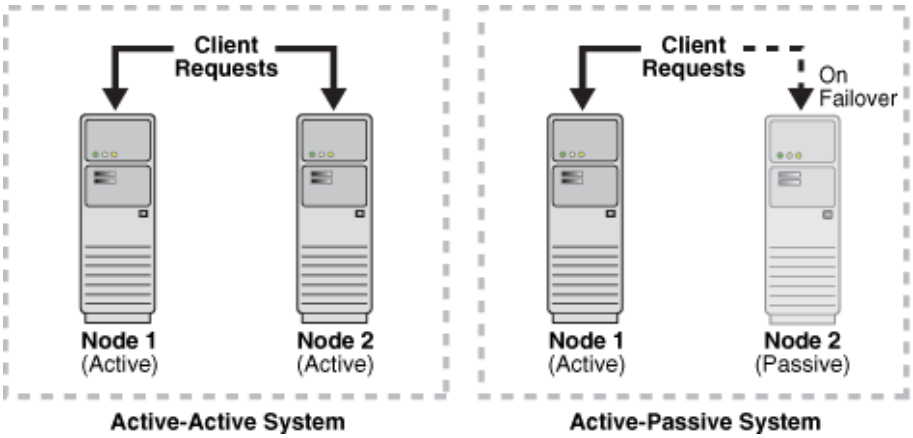


Obrázek 3.2: Schéma HA clusteru (Herbert, 2006)

3.2.1 Active-Active mód

Levá část obrázku 3.3 ukazuje active-active mód, kdy je poskytována část služeb clusteru na jednom uzlu a další část služeb na druhém uzlu. Pokud dojde k výpadku

například prvního uzlu, je druhý uzel schopný převzít navíc i služby z prvního uzlu. Výhodou je možnost rozložení zátěže mezi více uzlů. Je však potřeba mít na uzlech dostatek volných zdrojových prostředků pro případné převedené služby jiného uzlu.



Obrázek 3.3: Módy HA clusteru (Rich, 2011)

3.2.2 Active-Passive mód

Cluster v módu active-passive na pravé části obrázku 3.3 funguje tak, že veškerý provoz obstarává vybraný uzel (active node). Druhý uzel (passive node) pouze sleduje aktuální provoz a synchronizuje si stavy z aktivního uzlu. Neposkytuje žádné další služby clusteru. V případě výpadku aktivního uzlu je schopný jej okamžitě plně zastoupit. Tento stav má hlavní nevýhodu v tom, že jeden uzel je zcela bez využití až do doby, dokud nedojde k výpadku jiného uzlu.

3.2.3 Heartbeat

Jak již název napovídá jedná se o jakýsi pravidelný interval udržující cluster při životě. Implementace jsou různé. Z principu se jedná o pravidelný rytmus, ve kterém mezi sebou jednotlivé servery v clusteru komunikují a oznamují si, v jakém stavu se daný uzel nachází. Nejčastěji tato komunikace probíhá přes síťové připojení (nejlépe přímé propojení host-to-host) nebo je možnost propojením přes sériový port nebo v případech specializovaných komerčních řešení přes speciální sběrnici.

Pokud nějaký z uzlů v pravidelných intervalech neinformuje ostatní o jeho stavu, systém zjistí, že dochází v uzlu k nějakému problému. Dále zjišťuje, čím je chyba způsobena a pokusí se ji odstranit, případně informuje administrátora systému o nastávajícím problému.

3.2.4 Quorum

Tento výraz pochází z latiny, kde znamená kvantitativní podmínku pro platnost hlasování. V clusterovém prostředí se jedná o nastavitelnou část, která určuje, jak se systém bude rozhodovat, pokud dojde k nějakému problému. Podle definice clusteru,

kdy je definováno, že se jedná o skupinu počítačů, je zřejmé, že cluster musí obsahovat minimálně dva uzly.

Ale co nastane, pokud se jeden z těchto uzlů stane nedostupným? Například dojde k výpadku části počítačové sítě, která propojuje oba uzly. Jak systém pozná, že je chyba na straně prvního nebo druhého uzlu (nebo naopak), když mezi sebou navzájem uzly v daný okamžik nemohou komunikovat? První uzel si bude myslet, že je problém na straně druhého uzlu a druhý uzel si naopak bude myslet, že problém je na straně prvního uzlu. A v tomto okamžiku dojde k rozpadnutí clusteru a úplnému rozsynchronizování veškerých důležitých prvků. Celé by to mohlo vést k až k poškození uložených dat.

Zde přichází na řadu logika quorum, která řekne, jak má proběhnout hlasování o nedostupnosti ostatních uzlů. V případě dvou uzlů v clusteru bude quorum nastaveno tak, že cluster může považovat uzel za poškozený pouze v případě, že to o něm prohlásí dva uzly. A zde vzniká problém v tom, jak ve dvouuzlovém clusteru nastavit quorum. Jediným řešením je, že se do clusteru přidá ještě jeden uzel, aby byly tři a tím může hlasování skončit 2:1. Pokud chceme provozovat pouze dvouuzlový cluster, můžeme využít i takzvaný quorum disk. Jedná se např. o disk připojený iSCSI technologií. Na tento disk si oba uzly ukládají informace o svých stavech a pokud dojde na nějaký problém a dojde k remíze v hlasování, je quorum disk brán jako jakýsi rozhodující arbitr. Takto se například při výpadku sítě přesně lokalizuje, že závada je na straně jednoho z uzlů na cestě od síťové karty do přepínače. Tím pádem poškozený uzel není schopen komunikovat s ostatními zařízeními a je potřeba jej do odstranění problému z clusteru odebrat.

Některé implementace umožňují provozovat HA cluster bez nutnosti nastavení quorum stavu, ale pak je nutný v případě jakéhokoliv problému manuální zásah administrátora. Tím se výrazně prodlouží reakční doba na vyřešení problémové události a dojde ke ztrátě jedné z hlavní výhody HA clusteru – rychlé reakce na neočekávané výpadky.

3.2.5 Fencing

Další situace, která může nastat je, že uzly se nejsou schopny domluvit, na kterém uzlu jaké služby poběží. Například se nejsou schopny dohodnout, na kterém bude spuštěno virtuální prostředí pro nějaký konkrétní virtuální server. V případě kdyby do jednoho LVM svazku přistupovaly dva virtuální servery současně, došlo by k velice kritické situaci a nevratnému poškození filesystému virtuálního stroje a následné ztrátě jeho dat. Ochranu proti spuštění zcela identických služeb clusteru současně na různých uzlech pomáhá udržovat právě fencing.

Pokud nastane stav, že nějaký uzel drží službu, která byla přidělena jinému uzlu a nereaguje na příkazy, aby službu uvolnil, přijde na řadu mechanismus STONITH[1]. Tím dojde podle zvolené konfigurace například k tomu, že server je pomocí IPMI nebo iLO modulu restartován, nebo zcela vypnut. Jestliže server takovým modulem nedisponuje, je možné použít tzv. IP napájecí zásuvky, které jsou schopny vzdáleně zapínat a vypínat přívod elektrické energie do serveru.

Díky tomu je udržena integrita služeb a dat v clusteru.

[1] zkratka Shoot The Other Node In The Head

3.3 Hardware uzlů clusteru

Již při přípravě návrhu nového clusteru se setkáme s otázkou – jaký hardware použít pro uzly clusteru? Na výstavbu lze použít běžné konfigurace standardních serverů nebo dokonce i běžných osobních počítačů. Velice záleží na tom, co budeme po našem systému požadovat (např. výpočty, ukládání dat, virtualizaci).

Pokud navrhujeme vysoce dostupné řešení, je primární pamatovat na plnou redundanci systému. Je potřeba dostatečný počet síťových portů, a to minimálně 2 porty pro nutnou redundanci síťové komunikace. Je důležité si uvědomit, že pro HA cluster je životně důležitá komunikace mezi jednotlivými komponentami, která až na některé vyjímky, probíhá přes ethernetové rozhraní respektive počítačovou síť. Pokud nějaká část systému nekomunikuje a tváří se jako nedostupná už na vnitřní úrovni komunikace, je systémem vyhodnocena jako nefunkční a je okamžitě vyřazena z členství v clusteru.

Obecně doporučená minimální konfigurace pro uzel HA clusteru:

- serverová základní deska
- dva samostatné procesory
- paměť RAM s podporou ECC
- dvě nezávislá LAN rozhraní s rychlostí 1 Gb/s
- dva nezávislé napájecí zdroje
- diskový prostor zabezpečený RAID1

3.4 Počítačová síť pro cluster

Síťové propojení jak v rámci clusteru, tak propojení s veřejnými sítěmi je jedna z nejdůležitějších částí systému. Označují se také jako **interní** a **externí** síť clusteru. Interní síť nejčastěji slouží k vnitřní komunikaci (heartbeat) nebo připojení k SAN (Storage Area Network) zařízením. Externí sítě slouží ke komunikaci s uživateli, kterým jsou služby poskytovány. Nejčastěji jsou to připojení do sítě LAN nebo internetu. Obrázek 3.2 ukazuje schéma zapojení obou typů sítí.

Pokud je nespolehlivá interní síť, může celý cluster vykazovat neočekávané chyby, které se poměrně těžce lokalizují. Celý cluster se může dostávat do velice kritických stavů, které můžou vést až k úplnému rozpojení celého clusteru a jeho rozsynchronizování. Pokud je nespolehlivé spojení externí sítě, pak i v případě, že je celý cluster stabilní, není schopný uspokojovat požadavky uživatelů a stává se nedostupným.

Je velice důležitá i schopnost komunikace mezi jednotlivými síťovými přepínači (označované též jako *switch*). Z toho plyne, že pro clusterové sítě nelze používat nejlevnější řady přepínačů, které pouze přepínají síťový provoz. Je nutné použít přepínače vyšších řad, které již obsahují velmi sofistikovaný software pro řízení síťového provozu a jsou schopny vzájemné komunikace.

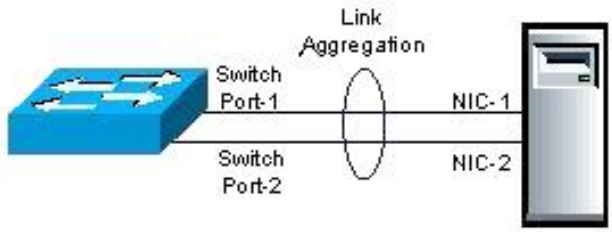
Redundance se v počítačových sítích vytváří pomocí takzvané agregace (slučování, kombinování). V praxi to znamená, že jeden uzel přípojíme více síťovými kabely do přepínače. Pokud bychom zapojili všechny kabely z jednoho uzlu do jednoho

přepínače, vzniklo by nám slabé místo. V okamžiku poruchy přepínače by i vícenásobné připojení uzlu nechránilo proti výpadku. Proto se každý kabel zapojí do různého přepínače, jak je znázorněno na obrázku 3.2. Aby síťové přepínače poznaly, že dané kabely pochází ze stejného uzlu, je nutné, aby si mezi sebou tyto informace vyměnily. K účelu agregace bylo vyvinuto několik metod. Níže jsou nejpoužívanější z nich.

Všechny zmíněné typy vytváří jakýsi virtuální kanál, který se skládá z jednotlivých fyzických propojů a automaticky je mezi ně rozkládána zátěž. Zároveň také zajišťuje redundanci tak, že v případě výpadku jednoho nebo i více z nich, použije ke komunikaci pouze funkční propoje.

3.4.1 LAG – IEEE 802.3ad

Zkratka LAG znamená *link aggregation group*. Jedná se o standard podle normy IEEE 802.1AX (z původního 802.3ad), který dnes vyžívají všichni velcí výrobci síťových přepínačů. Aby bylo možné tuto metodu využívat, je důležité, aby byl tento standard podporován přímo na úrovni hardware připojených zařízeních, které tímto protokolem komunikují. Je tedy nutné, aby přepínače měly přímo ve svém software zaintegrovánu podporu LAG od výrobce, stejně tak síťové karty ve svých firmware a ovladačích.



Obrázek 3.4: Zapojení LAG (Hammond, 2007)

Mód agregace může být buď **statický**, nebo **dynamický**. Statický typ funguje tak, že při prvotním nastavení uživatel vybere porty, které mají být součástí LAG (obrázek 3.4). Pokud ale dojde k výpadku jedné z fyzických linek, systém to nerozpozná a dojde k vysoké ztrátovosti síťových paketů, dokud administrátor problém manuálně nevyřeší. To je pro naší potřebu zcela nevyhovující. Pro HA cluster je vhodný LAG v dynamickém módu. Využívá *link aggregation control protocol* nebo-li zkráceně **LACP**.

Zjednodušeně popsat komunikaci tohoto protokolu tak, že se na porty nastavené v LACP módu vysílají speciální rámce LACPDU. Pokud se na druhé straně linky nachází zařízení, které také komunikuje pomocí LACP, začne odpovídat. Zároveň se pokusí detekovat další linky s tímto protokolem, které by mohly mezi těmito zařízeními existovat. Na základě toho se vytvoří virtuální kanál (virtuální linka). Na této virtuální lince dochází k pravidelné kontrole (keep-alive), zda jsou její jednotlivé fyzické linky funkční. Pokud nějaká fyzická linka selhala, je z virtuálního kanálu vyřazena až do doby opětovného korektního stavu.

Je důležité si uvědomit, jak probíhá u tohoto protokolu rozkládání zátěže. Tím, že vytvoříme například virtuální linku 2×1 Gb/s, negarantujeme, že zdvojnásobíme

přenosovou rychlost mezi dvěma zařízeními. Záleží na tom, jak komunikace probíhá. Pokud se jedná o datový tok do/z jedné konkrétní stanice, bude maximální rychlost stále 1 Gb/s. Provoz bude rozložen mezi více fyzických linek, když bude z různých klientů probíhat požadavek na jeden server připojený přes LAG. Pak je schopný tok dosáhnout v našem případě až 2 Gb/s průtoku.

Důvodem je to, že LAG udržuje konkrétní spojení vždy na jedné fyzické lince, protože pracuje na druhé vrstvě OSI a tedy vše probíhá na úrovni MAC adres, které nedokáží rozlišit IP adresu, číslo portu, druh protokolu atd. Existují i různá nestandardizovaná řešení, která dovedou rozložit zátěž i na vyšších vrstvách modelu OSI. Výhodou LAG je, že díky konkrétní komunikaci přes stejnou fyzickou linku jsou data přijata ve správném pořadí, jak byla odeslána a nevzniká tak žádná režie na poskládání správného pořadí nebo nutnosti opětovného poslání rámců.

Při vytváření LAG je nutné dodržet:

- stejná rychlost všech fyzických portů (např. 1 Gb/s)
- všechny porty ve full duplex módu (obousměrná komunikace)
- peer to peer zapojení (přímé propojení zařízení komunikujících mezi sebou pomocí LAG)

3.4.2 Round-robin

Jedná se o softwarovou implementaci sdružování portu v linuxovém prostředí v modulu *bonding driver*. Nepotřebuje podporu na straně přepínače. Vše se nastavuje na straně serveru. Funguje tak, že server postupně odesílá síťové pakety přes jednotlivé fyzické linky, které jsou do *bond* zařízení přidány – první paket na první linku, druhý paket na druhou linku a tak dále. Tato metoda si kontroluje stavy jednotlivých fyzických linek a díky tomu funguje i automatická redundance při výpadku nějaké z nich. Rovněž rozkládá zátěž, ale vzniká problém v tom, že poměrně často nastávají situace, kdy přijímající strana dostává pakety v jiném pořadí, než byly původně odeslány. To může mít za následek zpomalení sítě, protože některé fragmenty musí být poslány znovu. Správa bond zařízení probíhá pomocí programu *ifenslave*.

3.4.3 Active-backup

Opět jde o implementaci v linuxovém prostředí pomocí bonding modulu. Tato metoda nerozkládá zátěž, ale slouží pouze jako redundance propojení zařízení. Ve virtuální lince je aktivní pouze jeden fyzický link a další linky jsou v tzv. záložním módu. Pokud dojde k výpadku aktivního linku, tak jej zastoupí linka záložní.

3.5 Záložní zdroj – UPS

Neočekávaný výpadek napájení znamená pro HA cluster fatální událost, která má za následek okamžitou nedostupnost systému. Navíc při nekorektním vypnutí celého systému může dojít k nevratnému poškození neuložených dat. Proto je nutné celý cluster chránit před výpadkem elektrického proudu záložním zdrojem. Zkráceně se označuje také jako **UPS** podle anglického označení *uninterruptible power supply*.

Zálohování napájení se v praxi provádí dvěma způsoby. Pokud jsou servery umístěny v datovém centru, je záloha řešena na úrovni celého počítačového sálu respektive celé budovy. V případě výpadku proudu jsou veškeré servery napájeny z baterií, které jsou většinou v samostatné části budovy. Primárně tyto baterie slouží k vyrovnání krátkodobých výkyvů v elektrické síti. V případě dlouhodobého výpadku slouží jako mezičlánek, než nastartuje dieselový generátor, který je pak schopný zásobovat budovu elektrickým proudem dlouhodobě. Zde se tedy z pohledu administrátora předpokládá, že je zaručen nepřetržitý zdroj napájení.

Druhou možností je umístění clusteru například v rámci firemních prostor. Je tedy opět nutné zajistit ochranu systému před krátkodobými výkyvy elektrické sítě a v případě dlouhodobého výpadku zaručit clusteru dostatek času, aby se mohl korektně vypnout. Zde je nutné použít tzv. serverové nebo rackové UPS. Tyto záložní zdroje mají tu výhodu, že umožňují neustálou komunikaci s prvky clusteru a informují je o stavu napájení a případném výpadku. Díky tomu cluster reaguje na možný dlouhodobý výpadek a v případě kritického stavu napájení celý systém clusteru korektně ukončí.

Je velice důležité celý proces a logiku vypínání clusteru dokonale odladit už před uvedením do ostrého provozu, protože pak již není možné takové úpravy spolehlivě otestovat.

3.6 Clusterové operační systémy

Clusterové operační systémy byly z počátku doménou převážně velkých firem jako DEC, IBM, AT&T a další. Tyto firmy vyvíjely vlastní komerční řešení podle přání zákazníků. Cenové náklady byly velice vysoké. Postupně se přidávaly další společnosti s vlastním řešením. Díky široké nabídce komerčních i opensource řešení se v posledních deseti letech stal HA cluster dostupný i pro menší a střední firmy.

V současné době je několik velice známých operačních systémů vhodných pro implementaci vysoce dostupného clusteru. Každý z nich nabízí stejné základní služby clusteru. Hlavní rozdíly jsou v rozšiřujících službách, nástrojích pro ulehčení správy, ve vzhledu administračního rozhraní a jeho komfortu nebo rozsahu technické podpory.

Je velice dobré si při návrhu HA clusteru uvědomit, co je pro náš systém bezpodmínečně důležité, a co je naopak nepodstatné, protože od toho se odvíjí i správný výběr operačního systému clusteru.

3.6.1 OS Linux

Jedná se o operační systém, který je vyvíjen jako opensource, tedy je pro uživatele zdarma a jeho zdrojové kódy jsou veřejně přístupné. Vývoj tohoto systému je velice dynamický a spolupracuje na něm mnoho velkých firem i jednotlivců. Existuje mnoho

vydavatelů tohoto operačního systému, kde mezi nejznámější varianty patří Debian, RedHat, Ubuntu a další. Tato vydání se také nazývají **linuxové distribuce**, což je soubor velkého množství programů a jádra, které jsou navzájem odladěny a optimalizovány. Distribuci vždy zastřešuje buď nějaký spolek dobrovolníků, nebo komerční firma.

Převážná většina distribucí je zcela zdarma, v některých případech je možné zaplatit si podporu pro danou distribuci, což je velice žádané hlavně v případech, kdy se tento systém nasazuje například ve firemním prostředí. Pokud nějaká firma investuje do své IT infrastruktury peníze, požaduje za to nějakou záruku, že vybraný operační systém bude splňovat to, co sliboval a zároveň, že pokud dojde k nějakým problémům s vybraným operačním systémem, existuje někdo, kdo tento problém co nejdříve vyřeší.

Velkou výhodou tohoto operačního systému je jeho otevřenost, široká podpora výrobců hardwaru i důležitých korporací v oboru, velká komunita vývojářů a široká podpora při řešení problémů. Dalšími výhodami jsou rychlost vydávání bezpečnostních aktualizací, široká podpora různých platforem a opravdu velký výběr aplikací (pro představu poslední verze distribuce Debian obsahuje skoro 60 000 balíčků).

Díky své otevřenosti tento systém láká řadu nových vývojářů, a tak vzniká mnoho nových aplikací a rozšíření. Mezi ně patří i rozšíření pro HA cluster zvané **Proxmox**, které je primárně určeno pro distribuci **Debian Linux** a na kterém bude realizována praktická část této práce.

3.6.2 Microsoft Windows Server

Produkty tohoto softwarového gigantu jsou všeobecně známé. Před řadou let se vedení společnosti domnívalo, že má tato firma neotřesitelnou pozici nejen na běžných počítačích, ale i na poli serverů.

Z důvodu dynamického vývoje konkurenčních systémů docházelo postupně k vytlačování tohoto systému z některých aplikačních prostředí a nahrazování konkurenčními alternativami. Vedení firmy si začalo uvědomovat důležitost serverové části trhu a vyvinulo velice kvalitní operační systém Windows Server 2008. S ním přišla i důležitá podpora vysoce dostupného serveru (zde pojmenována jako failover cluster). V každé následující verzi Windows Server dochází vždy k nějakému vylepšení a dnes je HA cluster v tomto operačním systému na velice dobré úrovni. Nevýhodou tohoto řešení je vysoká pořizovací cena.

3.6.3 vSphere

vSphere je produkt firmy VMware, která patří mezi průkopníky na poli virtualizace. Jako jedna z prvních zjistila, že technologie virtualizace a clusterů jdou velice dobře zkombinovat dohromady a dosáhnout tím na tehdejší dobu neuvěřitelných výsledků. V podstatě tak stála u zrodu současných clusterových a cloudových řešení. Některé ze svých produktů poskytuje bezplatně, ovšem za cenu určitých omezení, což může být hlavně u komerčních řešeních nepřijatelné.

Produkty s podporou HA clusteru jsou pouze placené, a proto je potřeba pořídit odpovídající licence produktu vSphere.

3.7 Clusterové úložiště

Další důležitou částí každého clusteru jsou jeho datová úložiště – **storages**. Ukládají se na ně jak uživatelská data, tak mohou být využita k uložení systémových částí. Hlavním rozdílem těchto úložišť, oproti standardnímu použití, je schopnost rozlišovat, kdo k daným datům aktuálně přistupuje. V případě nehlídání takového stavu mohou nastat situace, kdy by jednotlivé uzly clusteru mohly zapisovat ve stejný čas do stejného místa na úložišti. To je velice kritická událost vedoucí k poškození ukládaných dat nebo celých souborových systému pro virtuální počítače.

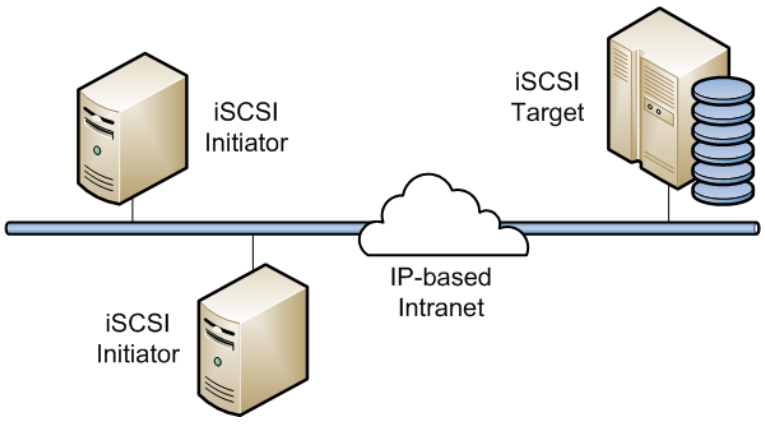
U běžných serverů je možně umístit úložiště dat přímo v serveru samotném. U clusteru je to složitější, protože je potřeba stejná a aktuální data sdílet s ostatními uzly clusteru, aby v případě výpadku jednoho z uzlu mohl jeho funkci zastoupit jiný.

Řešení tohoto problému jsou dvě. Využívat nějaké sdílené síťové úložiště, nebo veškerá data okamžitě replikovat mezi všemi uzly clusteru. Pro sdílená úložiště je nejčastějším komunikačním protokolem iSCSI protokol.

3.7.1 iSCSI

Zkratka vychází z anglického názvu Internet *Internet Small Computer Systems Interface*. Jedná se o protokol, který umožňuje připojovat disková pole pomocí počítačové sítě. Je postavený na protokolu TCP/IP a navržený tak, aby si poradil i s případnými latencemi sítě i možnými krátkodobými výpadky. Samozřejmě je optimální se takovým problémům vyhnout, a proto se pro infrastrukturu velice často používá technologie redundantních optických vláken. Levnější a častější je softwarová implementace, kdy na straně serveru (diskového pole) je spuštěna služba označena jako **iSCSI Target** a na straně klienta (uzlu) běží takzvaný **iSCSI Initiator**.

Celé zapojení názorně ukazuje obrázek 3.5. Zařízení připojené iSCSI protokolem se v daném uzlu projevuje, jako by se jednalo o lokálně instalovaný disk a operační systém s ním i tak pracuje.



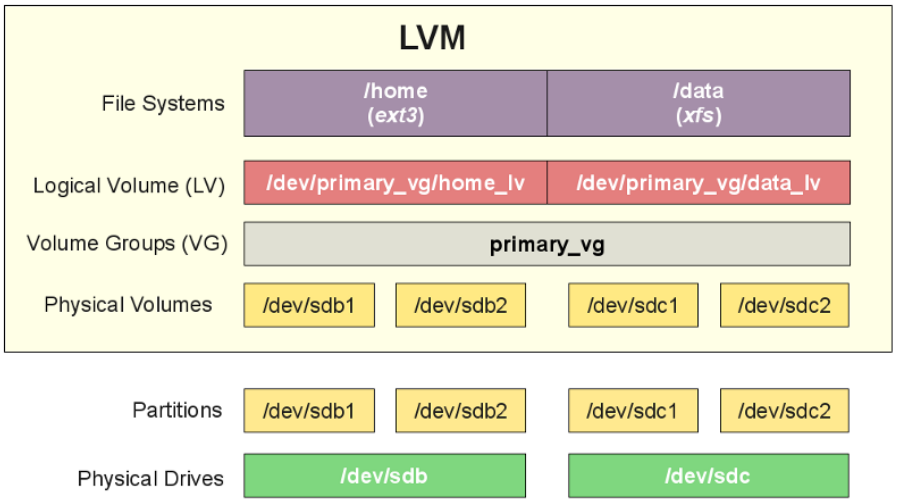
Obrázek 3.5: iSCSI (Administrator's Guide for Release 6, 2016)

3.7.2 LVM

Zajímavou možností pro systém ukládání dat na diskových polích je *Logical Volume Manager*, zkráceně **LVM**. Klasické dělení disku pomocí programu *fdisk* je široce známým postupem, jak rozdělit běžné disky. Tento systém je vhodný pro operační systém a případná uživatelská data na konkretním serveru, ale je nedostačující pro potřeby diskových polí, kde mohou být velice různorodá a rozlehlá data. U velkých úložišť se poměrně špatně odhaduje, jak bude rychlý nárůst ukládaných dat, nelze pole odstavit kvůli záloze určitých dat apod. Z tohoto pohledu je velice důležitá flexibilita práce s prostorem na diskovém poli.

LVM je implementace z operačního systému Linux, ale také ostatní platformy mají své vlastní řešení jako například dynamické svazky a shadow copy v systémech Microsoft. LVM mimo jiné umožňuje dynamicky pracovat se stávajícími svazky (zvětšovat, zmenšovat), vytvářet nové, slučovat jednotlivé diskové prostředky do jednoho virtuálního celku a dělat takzvané snapshoty, které jsou vhodné pro zálohování systému. Snapshot (snímek) je jakési zastavení v čase, kdy se data k danému času takzvaně zmrazí a od té chvíle nedochází na nově vzniklém snapshotovém svazku k jejich změně. To je výhodné hlavně pro zálohování systému, protože data se během provádění záloh nemění a zůstávají ve stavu k přesnému časovému okamžiku. Nejdůležitější je fakt, že veškeré tyto operace lze provádět za plného běhu diskového pole, kde je tato technologie použita. Z toho vyplývá, že je vždy nutné tyto změny nejprve řádně připravit a rozmyslet.

Pro potřeby clusteru vzniklo rozšíření cLVM – *Cluster Logical Volume Manager*. Jedná se o službu, která ve spolupráci s LVM zajišťuje správné zamykání jednotlivých svazků jako ochranu před případnými současnými přístupy z různých uzlů clusteru.



Obrázek 3.6: Základní části LVM (Layton, 2010)

Popis obrázku 3.6:

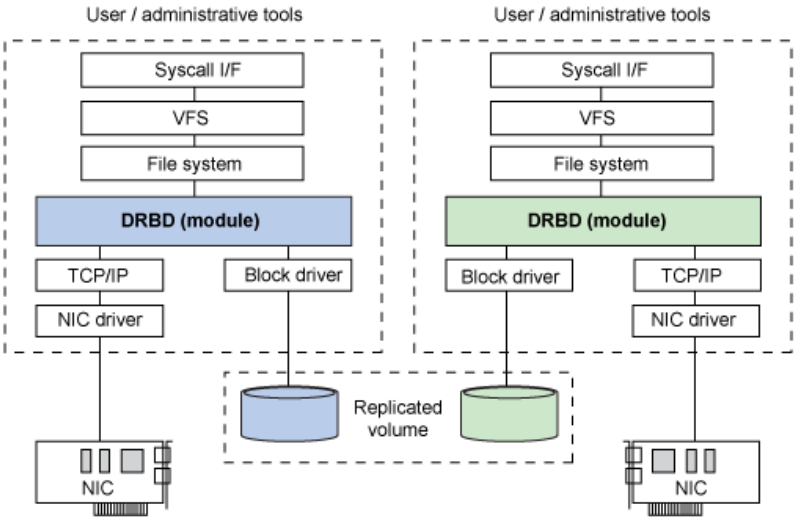
- **Physical Volume – PV**: jedná se o disk nebo diskový oddíl, který je připraven pro práci s LVM (lze chápat jako blokové zařízení)
- **Volume Groups – VG**: je skupina fyzických oddílů – jsou zde přiřazeny jednotlivé PV (podobné fyzickému disku)

- **Logical Volume – LV**: je finální logický svazek, se kterým lze pracovat jako s běžným svazkem (podobné konkrétnímu diskovému oddílu)

3.7.3 DRBD

Distributed Replicated Block Device (DRBD) je technologie operačního systému Linux, která umožňuje synchronizaci dat v reálném čase na diskových jednotkách mezi různými uzly prostřednictvím počítačové sítě a TCP/IP protokolu. Zjednodušeně ji můžeme nazvat jako RAID1 přes počítačovou síť, jak je patrné z obrázku 3.7. Na připojených blokových zařízeních zajišťuje stejná relevantní data ve stejném okamžiku. Tím je zaručena distribuce a redundance dat mezi jednotlivými uzly clusteru.

Hlavní výhodou DRBD je, že není nutné používat velice drahé redundantní SAN řešení pomocí samostatných jednotek. Je možné vytvořit disková pole přímo na jednotlivých uzlech clusteru a ty mezi sebou synchronizovat pomocí DRBD replikace. Původní návrh DRBD předpokládal replikaci pouze mezi dvěma uzly, ale poslední verze (9.x) podporují až 16 uzlů. Další velkou výhodou je schopnost geograficky vzdálené replikace, kdy je možné v reálném čase udržovat stejná data v různých vzdálených lokalitách.



Obrázek 3.7: Architektura DRBD (Jones, 2010)

3.7.4 NFS

Network File System (NFS) je jedním z nejstarších a nejpoužívanějších síťových souborových systémů pro sdílení dat mezi servery a stanicemi. Standardně ke své funkci využívá protokol UDP (novější verze podporují i TCP). Jeho nevýhodou je jeho relativní nebezpečnost (pro použití mimo uzavřené sítě), a to hlavně z důvodu základu na technologii RPC. NFS prošel dlouholetým vývojem a dnes je dostupný ve verzi 4, která jej mimo jiné rozšířila o důležitou součást větší bezpečnosti (kerberos). Další

nevýhodou je poměrně malý výkon při velkém zatížení, kdy v okamžiku mnoha požadavků jeho rychlost výrazně klesá. Nehodí se pro každou implementaci a je nutné vždy posoudit vhodnost nasazení v daném prostředí. Využívá se například pro ukládaní záloh z clusteru.

3.7.5 GFS2

Souborový systém *Global File System* je přímo navržený pro práci v clusteru. Jeho vývoj zaštiťuje firma RedHat, která jej využívá ve vlastním clusterovém řešení (RedHat Enterprise Linux). Nejčastěji je tento souborový systém využíván v kombinaci s iSCSI, kdy je připojen sdílený iSCSI disk k jednotlivým uzlům clusteru současně a až na úrovni systému souborů jsou řešena zamykání proti současnému přístupu pomocí distribuovaného správce zámků (DLM). Všechny uzly jsou si rovny, neexistuje žádná serverová část. Výhodou je zachovaný vysoký výkon při více uzlech (klientech) a jednoduchá škálovatelnost.

3.7.6 CSV

Cluster Shared Volumes je funkční rozšíření souborového systému NTFS od firmy Microsoft, který byl prvně uveden ve Windows Server 2008 R2. Umožňuje vytvářet sdílené svazky pro vysoce dostupné řešení na systému Windows Server. Toto rozšíření je určené pro zamykání VHD souborů virtuálních počítačů uložených na jednom svazku. VHD jsou pak korektně zpřístupněny na různých uzlech clusteru. Jedná se tedy o poměrně specifické použití pouze u vysoce dostupného clusteru pro virtuální počítače.

3.7.7 VMFS

Virtual Machine File System pochází od firmy WMware a je určen pro provoz virtuálních serverů v prostředí VMware HA clusteru. Slouží k ukládání virtuálních disků jednotlivých virtuálních strojů a umožňuje jejich korektní provoz v rámci clusteru. V současné době je dostupný ve verzi 5, kde může být celková velikost prostoru až 64 TB a maximální velikost virtuálního disku může být až 62 TB.

3.8 Virtualizace

Slovem virtualizace se v IT označuje mnoho různých technologií k řešení daných situací. Zde je pod pojmem virtualizace myšlena celá virtualizace, neboli virtuální stroj (**VM** – virtual machine). Hardware s operačním systémem, na kterém je spuštěno prostředí umožňující běh virtuálních strojů (**hypervisor**), se nazývá anglickým slovem **host**, česky hostitel. Jednotlivé virtuální stroje se nazývají **guest**, česky host.

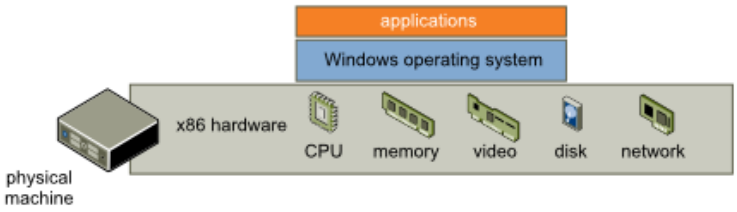
Způsobů, jak poskytnout virtualizované prostředí pro virtuální stroj, je více. V dnešní době jsou nejčastěji využívány technologie plné virtualizace, paravirtualizace a virtualizace na úrovni operačního systému (kontejnerová).

Plná virtualizace je prostředí, kde jsou virtualizovány veškeré prostředky pro hostovaný operační systém. Je nutná hardwarová podpora přímo v procesoru. Celé prostředí je plně virtualizováno (procesor, paměť, grafická karta, řadič pevného disku, síťová karta atd.) a není nutná žádná úprava hostovaného operačního systému. Tento způsob je poměrně náročný na prostředky hostitele, ovšem v dnešní době je mnoho komponent serverů pro virtualizaci hardwarově optimalizováno. Hostovaný systém prakticky nepozná, že je spuštěný ve virtualizovaném prostředí.

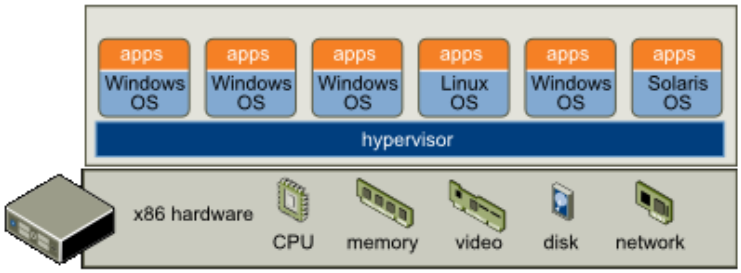
Paravirtualizace je částečná virtualizace. To znamená, že část systému může být virtualizována, ale určitá část prostředí komunikuje pomocí API rozhraní přímo s hypervisorem. Nejvíce se využívá u I/O operací, které jsou při plné virtualizaci výrazně pomalejší než přímo na samotném hostiteli. Tato technologie vyžaduje zásah do hostovaného operačního systému nejčastěji formou ovladače zařízení (například řadič disku, síťová karta). Díky tomu lze dosahovat výrazně vyššího výkonu u vstupních a výstupních operací. Hostovaný systém s ohledem na nutnou úpravu má informace o tom, že je spuštěný ve virtuálním prostředí.

Kontejnerová virtualizace Jedná se o jiný přístup k možnosti virtualizovaného prostředí. Veškeré virtuální stroje využívají stejné jádro operačního systému jako hostitel, ale aplikace běží v izolovaném prostředí. Tyto aplikace vnímají vyhrazený izolovaný prostor jako nový systém. Z toho vyplývá, že host musí používat identický operační systém jako hostitel, resp. operační systém hostovaného systému musí být schopen korektně fungovat s jádrem hostitelského serveru. Výhodou této technologie je minimální režie na běh virtuálních strojů, protože v podstatě dochází pouze k izolaci mezi procesy jednotlivých virtuálních strojů.

Na obrázku 3.8 je znázorněno, jak vypadá provoz aplikací a služeb na běžném nevirtualizovaném serveru. Obrázek 3.9 zobrazuje fyzický server poskytující prostřednictvím hypervizoru virtuální prostředí pro několik virtuálních strojů.



Obrázek 3.8: Nevirtualizovaný server (Austin, 2012)



Obrázek 3.9: Virtualizovaný server (Austin, 2012)

Vzniká otázka – proč používat virtualizaci, když je zcela evidentní, že jsou zde v některých případech poměrně vysoké nároky na režii a část výkonu našeho hardware

zatížíme emulováním prostředí pro virtuální stroje? V počátcích virtualizace byl pokles výkonu velice znatelný (v řádu desítek procent). Dnes významní výrobci procesorů, řadičů, základních desek a dalších komponent berou virtualizaci jako standard, který musí svým zákazníkům nabídnout, a tak je s její podporou počítáno již v samotném návrhu hardware. Tím došlo k výraznému snížení nároku na režii (v řádu několika jednotek procent).

Samozřejmě je důležité rozlišovat, jaký systém budeme virtualizovat. Například server, který bude neustále pod vysokou zátěží veškerých svých komponent nemusí být zcela efektivní převést do virtuálního prostředí. Virtualizace nám ale nabízí mnohem více výhod. Zde jsou některé z nich:

- úspora nákladů na pořízení HW (výrazně efektivnější využití HW),
- úspora elektrické energie,
- flexibilita systému (rozšíření operační paměti, diskového prostoru apod.),
- zjednodušení správy (jednoduché zálohování, přesun virtuálního stroje atd.),
- provoz více operačních systémů na jednom fyzickém serveru.

4. Řešení a výsledky

V této časti bude popsaný konkrétní postup, jak realizovat implementaci vysoce dostupného clusteru. Operačním systémem clusteru bude **Debian Linux 7** s rozšířením **Proxmox**. Již výchozí distribuce OS Debian v základu obsahuje nástroje pro vytvoření počítačového clusteru, nicméně rozšíření Proxmox nabízí výrazně propracovanější rozhraní pro správu clusteru a upravuje systém tak, že je optimalizovaný pro běh v clusterovém prostředí. Zároveň vývojáři tohoto rozšíření nabízí placenou technickou podporu s garantovanou dobou pro řešení případných problémů, což je velice důležité pro nasazení například ve firemním prostředí. Rozšíření je určeno pro vytváření vysoce dostupného clusteru poskytujícího prostředí pro virtuální stroje.

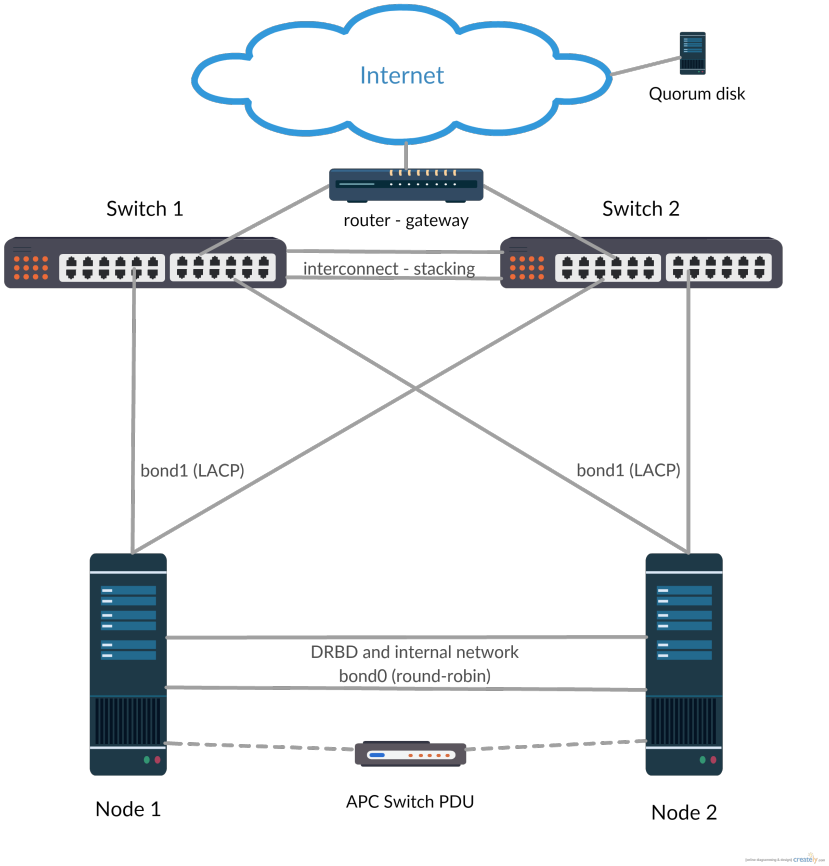
Konkrétní doporučený hardware pro reálný provoz je detailně popsán v příloze A na straně 43, kde jsou k porovnání i pořizovací ceny HA clusteru a klasického systému s jedním serverem. Stejnou konfiguraci osobně používám ke komerčním účelům více jak dva roky. Samozřejmě vždy záleží na tom k čemu budeme cluster využívat a podle toho zvolíme odpovídající hardware. Pro potřeby této práce postačují výrazně levnější komponenty, které jsou vypsány níže, případně i s jejich základními parametry:

- 2×počítač (1×CPU s podporou virtualizace, 4 GB RAM, 3×LAN, 2×HDD),
- 2×switch Netgear GS724TS (zapojené do stacku),
- 1×virtuální server poskytující iSCSI disk pro quorum,
- 1×modul řídící napájení APC Switch Rack PDU,
- 1×router TP-LINK TL-WDR4300 (firmware OpenWRT),
- UTP kabely s koncovkami RJ45.

Obrázek 4.10 ukazuje zapojení clusteru. Každý uzel ukládá data na softwarový RAID1 a disponuje třemi síťovými kartami. Uzly jsou nápájeny prostřednictvím APC PDU kvůli případnému použití fencigu, tzn. možnost vypnutí napájení vybraného serveru. Uzly jsou mezi sebou propojeny přímo kabelem, který slouží k replikaci pomocí DRBD modulu a ke komunikaci v rámci clusteru. Tento propoj je připravený pro práci v agregaci tzv. bondu (bond0). Pro testovací účely je použita pouze jedna síťová karta na každé straně, ale pro reálný provoz důrazně doporučuji alespoň dva porty na každé straně. Připojení jednotlivých uzlů do sítě umožňuje další agregovaná linka (bond1) do přepínačů. Vždy jeden kabel ze stejného uzlu do jiného přepínače. Tím je zaručena redundance síťového připojení. Přepínače jsou mezi sebou propojeny tzv. stackovacím kabelem. Stacking přepínačů je jakási virtualizace, kdy můžeme z více menších přepínačů vytvořit jeden velký virtuální, který pak spravujeme. Důležitou

výhodou vybraných přepínačů je, že zvládají tzv. automatický failover. Pokud jeden z nich zkolabuje, druhý je schopný zachovat v paměti stejnou konfiguraci (stát se master switchem), a tak udržet síť funkční. Toto je velice důležitý prvek, protože pokud by přepínače mezi sebou nekomunikovaly, nebyla by možná redundance pomocí LACP.

Protože implementujeme pouze dvouuzlový cluster, je potřeba vytvořit ještě tzv. quorum disk. Ten zprostředkuje virtuální stroj pomocí iSCSI a pro uzly bude přístupný skrze nadřazený router, tedy není ve stejné síti jako cluster. Virtuální server nemusí být nutně v internetu, ale může být například na jiném segmentu sítě. V reálném provozu by bylo vhodné data přenášená z internetu přes iSCSI ještě šifrovat, ale protože zde nejsou žádné citlivé informace (pouze informace o stavu uzlů), postačí zabezpečení prostřednictvím hesla a nastavením firewallu.



Obrázek 4.10: Schéma implementovaného clusteru (autor)

Síťové nastavení v clusteru:

- doména test.lan
- node1.test.lan: bond0: 192.168.100.10/24; bond1: 192.168.1.10/24
- node2.test.lan: bond0: 192.168.100.20/24; bond1: 192.168.1.20/24
- switch swt.test.lan : 192.168.1.100/24
- APC Switch Rack PDU pdu.test.lan: 192.168.1.101/24
- adresa hraničního routeru (gateway), nameserver: 192.168.1.1

4.1 Instalace OS Debian Linux

1. Stažení instalačního média:

   ```
   wget http://cdimage.debian.org/mirror/cdimage/archive
      /7.9.0/amd64/iso-cd/debian-7.9.0-amd64-netinst.iso
   ```

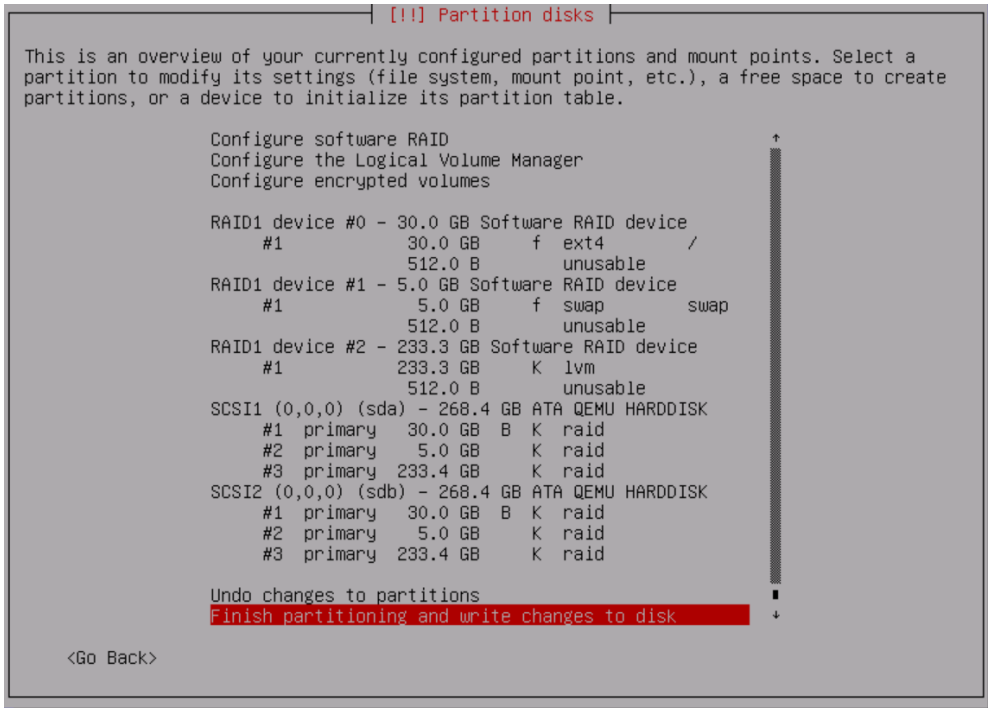
2. Instalace bude probíhat z USB flash disku (zařízení /dev/sdb):

   ```
   cp debian-7.9.0-amd64-netinst.iso /dev/sdb
   ```

   Instalační médium je připravené, můžeme instalovat první uzel.

3. Spustíme instalaci z USB flash disku na prvním uzlu.
4. Vybereme jazykové prostředí pro instalaci *English*.
5. Vybereme pásmo, ve kterém se server nachází *Europe – Czech Republic*.
6. Nastavíme systémové proměnné pro locale *en_US.UTF-8*.
7. Zvolíme rozložení klávesnice *American English*.
8. Konfigurace sítě *Configure network manually*.
9. IP adresa *192.168.1.10*, maska *255.255.255.0*, brána *192.168.1.1*, nameserver *192.168.1.1*.
10. Název serveru *node1*, patří do domény *test.lan*.
11. Nastavíme heslo uživatele *root* a vytvoříme prvního uživatele *test*.
12. Na obou pevných diskách vytvoříme tři oddíly: 30GB pro systém, 5GB pro swap a zbytek pro virtuální stroje (DRBD).
13. Vybereme *Configure software RAID* a zde nastavíme tři pole s RAID1 viz obrázek 4.11.
14. Zvolíme lokální zrcadlo zdroje balíčků *Czech Republic – ftp.cz.debian.org*.
15. U výběru software, který má být instalovaný necháme vše prázdné – postačuje nám úplný základ systému.
16. Zavaděč systému GRUB necháme nainstalovat do master boot record.

Tím je instalace dokončena. Stejný postup opakujeme i na druhém uzlu s tím, že upravíme na název a IP adresy pro něj určené.



Obrázek 4.11: Rozdělení disků a vytvoření RAID (autor)

4.2 Clusterová nadstavba Proxmox

Máme připravený čistý operační systém na obou uzlech. Nyní je potřeba systém připravit pro instalaci rozšíření Proxmox, které nám umožní vytvořit HA cluster.

1. Nejprve doinstalujeme do systému nějaké základní balíčky, aby se nám s ním lépe pracovalo:

   ```
   apt-get install screen less mc openssh-server
   ```

2. Zajistíme schopnost systému bootovat z obou dvou disků tak, že přidáme /dev/sda a /dev/sdb jako místo, kam se má instalovat zavaděč GRUB:

   ```
   dpkg-reconfigure grub-pc
   ```

3. Do adresáře */etc/apt/source.list.d/* přidáme soubor *proxmox.list*, který bude obsahovat:

   ```
   deb http://download.proxmox.com/debian wheezy pve-no-
      subscription
   ```

4. Je nutné přidat důvěryhodný klíč repozitáře, aby instalované balíčky měly podpis:

```
wget -O- "http://download.proxmox.com/debian/key.asc" |
   apt-key add -
```

5. Zaktualizujeme balíčky pocházející z přidaného repozitáře:

   ```
   apt-get update && apt-get dist-upgrade
   ```

6. Nainstalujeme nová jádra optimalizovaná pro Proxmox. Jádro 3.10 má lepší podporu KVM, ale neobsahuje modul pro kontajnerovou virtualizaci OpenVZ, kterou používat nebudeme:

   ```
   apt-get install pve-firmware pve-kernel-2.6.32-44-pve pve-
      kernel-3.10.0-16 && reboot
   ```

7. Odebereme nepotřebné jádro a další balíčky:

   ```
   apt-get remove linux-image-amd64 linux-image-3.2.0-4-amd64
       linux-base && update-grub
   ```

8. Nyní je systém připravený k instalaci rozšíření Proxmox. Spustíme tedy instalaci:

   ```
   apt-get install proxmox-ve-2.6.32 ntp ssh lvm2 postfix ksm
      -control-daemon vzprocps open-iscsi bootlogd
   ```

   Po restartu je již přístupné webové rozhraní pro správu Proxmox na adrese:

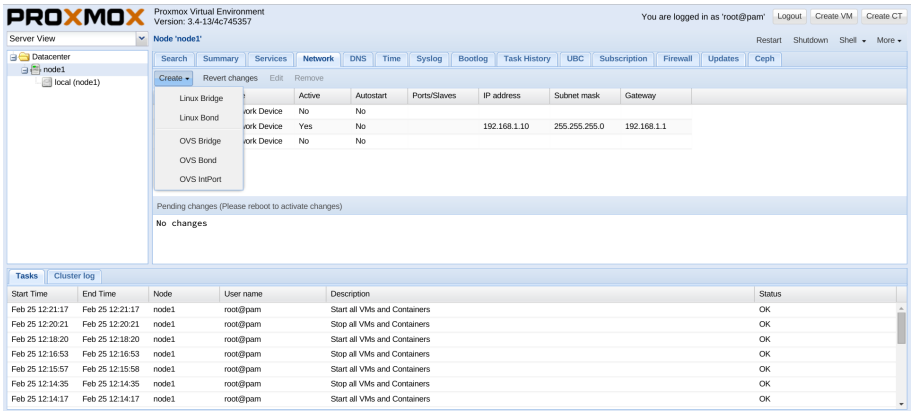
   ```
   https://192.168.1.10:8006
   ```

9. To samé provedeme na druhém uzlu.

4.3 Konfigurace sítě

Základní operační systém a Proxmox je na uzlech nainstalovaný. Nyní je potřeba správně nastavit síťová rozhraní, jak je popsáno v úvodu. Nastavení můžeme provést pomocí editace konfiguračního souboru */etc/network/interfaces* nebo přes webové rozhraní (obrázek 4.12).

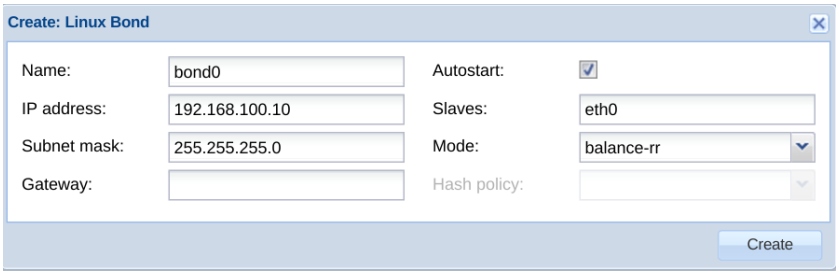
1. Přihlásíme se přes prohlížeč: *https://192.168.1.10:8006*
2. V levém panelu vybereme *node1* – v hlavním okně záložku *Network*.



Obrázek 4.12: Proxmox – webové rozhraní (autor)

3. Smažeme aktuální konfiguraci IP adres pro rozhraní, přes které jsme byli provizorně připojeni během instalace (např. pro eth1).

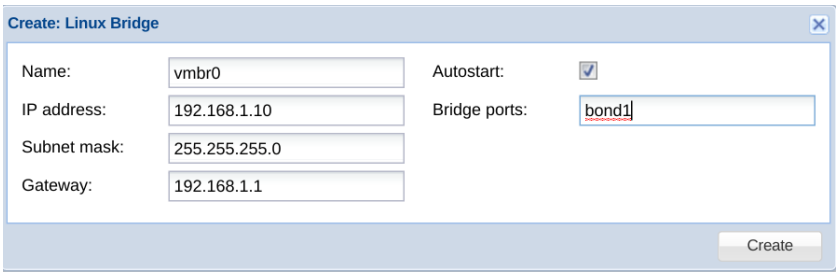
4. Vytvoříme nové bond0 a bond1 síťové rozhraní – *Create – Linux Bond*:



Obrázek 4.13: Proxmox – vytvoření bond zařízení (autor)

Pro bond1 nevyplňujeme IP adresu, masku, gateway. Pouze *slaves* obsahuje *eth1 eth0* a *mode* je nastaveno na *LACP (802.3ad)*.

5. Vytvoříme bridge zařízení vmbr0, které bude obsahovat bond1 – *Create – Linux Bridge*:



Obrázek 4.14: Proxmox – vytvoření bridge zařízení (autor)

6. upravíme soubor */etc/hosts*:

```
127.0.0.1 localhost
192.168.100.10 node1.test.lan node1
192.168.100.20 node2.test.lan node2
```

7. Zrestartujeme uzel pro zavedení nového nastavení sítě.
8. Na přepínači na portech, kde je uzel připojen vytvoříme sdružené porty LACP.
9. Stejný postup použijeme i na druhý uzel. Tím máme připravenou počítačovou síť pro provoz clusteru.

4.4 Konfigurace DRBD

Jako clusterové datové úložiště budeme využívat lokální disky v jednotlivých uzlech, které se budou synchronizovat pomocí jádrového modulu DRBD. Pro tato data je vyhrazena největší část diskového prostoru na RAID1 poli (zařízení /dev/md2).

1. Přidáme repozitář backports – do souboru */etc/apt/sources.list* přidáme řádek:

```
deb http://ftp.cz.debian.org/debian/ wheezy-backports main
```

2. Doinstalujeme aktuální balíček pro správu DRBD (předtím odebereme repozitář pro placenou podporu a zaktualizujeme seznam možných balíčků k instalaci):

```
rm /etc/apt/sources.list.d/pve-enterprise.list && apt-get
   update && apt-get -t wheezy-backports install drbd8-
   utils
```

3. Upravíme obsah souboru */etc/drbd.d/global_common.conf* na:

```
global { usage-count no; }
common {
      syncer { rate 100M; verify-alg md5; }
      handlers { out-of-sync "/usr/lib/drbd/notify-out-of-
         sync.sh␣root"; }
}
```

4. Přidáme soubor */etc/drbd.d/r0.res*, který obsahuje:

```
resource r0 {
      protocol C;
      startup {
            wfc-timeout 0;
```

```
            degr-wfc-timeout 60;
            become-primary-on both;
        }
        net {
            cram-hmac-alg sha1;
            shared-secret "oogohZ9aReelif6Jee9Ohghohnohch"
               ;
            allow-two-primaries;
            after-sb-0pri discard-zero-changes;
            after-sb-1pri discard-secondary;
            after-sb-2pri disconnect;
        }
        on node1 {
            device /dev/drbd0;
            disk /dev/md2;
            address 192.168.100.10:7788;
            meta-disk internal;
        }
        on node2 {
            device /dev/drbd0;
            disk /dev/md2;
            address 192.168.100.20:7788;
            meta-disk internal;
        }
    disk {
        no-disk-barrier;
        no-disk-flushes;
    }
}
```

5. Na obou uzlech spustíme démona drbd:

   ```
   /etc/init.d/drbd start
   ```

6. A na obou uzlech založíme na replikovaném zařízení část pro metadata a spustíme replikaci dat:

   ```
   drbdadm create-md r0 && drbdadm up r0
   ```

7. Na jednom z uzlů spustíme příkaz:

   ```
   drbdadm -- --overwrite-data-of-peer primary r0
   ```

   Tím určíme, že tento uzel má být brán jako primární pro prvotní replikaci.

8. Po zadání příkazu:

   ```
   cat /proc/drbd
   ```

   Vypíše systém něco podobného jako:

   ```
   version: 8.4.3 (api:1/proto:86-101)
   srcversion: 19422058F8A2D4AC0C8EF09
    0: cs:SyncSource ro:Primary/Secondary ds:UpToDate/
       Inconsistent C r-----
      ns:6309888 nr:0 dw:0 dr:6314648 al:0 bm:384 lo:7 pe:0
         ua:8 ap:0 ep:1 wo:d oos:271942128
         [>....................] sync'ed: 2.3%
            (265568/271728)Mfinish: 1:39:21 speed: 45,616
            (26,848) K/sec
   ```

9. Na obou uzlech upravíme soubor */etc/lvm/lvm.conf* tak, aby se LVM svazky hledaly pouze na novém zařízení */dev/drbd0*:

   ```
   filter = [ "a|drbd0|", "r|.*|" ]
   ```

10. Založíme nový fyzický LVM svazek (tento a následující příkazy zadáváme pouze na jednom uzlu – replikace je již funkční):

    ```
    pvcreate /dev/drbd0
    ```

11. Vytvoříme volume group pro virtuální počítače, kterou pojmenujeme vm:

    ```
    vgcreate vm /dev/drbd0
    ```

Prvotní replikace dat trvá delší dobu (podle velikosti oddílu), protože se přenášejí veškerá data z jednoho uzlu na druhý.

Pokud se po kompletní replikaci po zadání příkazu:

```
cat /proc/drbd
```

neobjeví status

```
ro:Primary/Primary
```

je potřeba zrestartovat DRBD démon na obou uzlech příkazem:

```
/etc/init.d/drbd restart
```

4.5 Zprovoznění HA clusteru

Infrastruktura clusteru je připravena, nyní je důležité nakonfigurovat systém, aby se jako HA cluster choval.

1. Nastavíme název clusteru – například na uzlu node1:

   ```
   pvecm create cluster-test
   ```

2. Zkontrolujeme informace o vytvořeném clusteru – node1 by měl být automaticky přidaný jako člen clusteru:

   ```
   pvecm status
   ```

   Tento příkaz nám ukazuje stav clusteru a další důležité podrobnosti.

3. Z konzole na node2 přidáme do clusteru i tento uzel (zadává se IP adresa uzlu, který je již v clusteru, resp. odkud si má nový uzel vzít informace o clusteru):

   ```
   pvecm add 192.168.100.10
   ```

4. Uzly, které cluster obsahuje, zobrazíme příkazem:

   ```
   pvecm nodes
   ```

5. Tím máme základ clusteru připravený. Protože máme pouze dva uzly, je potřeba kvůli správným rozhodovacím procesům (quorum) připojit quorum disk. Upravíme řádky souboru */etc/iscsi/iscsid.conf* na obou uzlech následovně:

   ```
   discovery.sendtargets.auth.authmethod = CHAP
   discovery.sendtargets.auth.username = quorumdisk
   discovery.sendtargets.auth.password = bezpecneheslo
   node.session.auth.authmethod = CHAP
   node.session.auth.username = quorumdisk
   node.session.auth.password = bezpecneheslo
   ```

6. Zjistíme si, jaké jsou k dispozici iscsi jednotky (na obou uzlech):

   ```
   iscsiadm -m discovery -t st -p 160.217.161.169
   ```

7. A připojíme jednotku určenou pro quorum disk (na obou uzlech):

   ```
   iscsiadm -m node --targetname "iqn.2016-02.lan.test.quorum
      :qdisk" --portal "160.217.161.169:3260" --login
   ```

8. Na jednom z uzlů quorum disk připravíme k použití – vytvoříme klasický svazek:

```
fdisk /dev/sdc
```

9. A naformátujeme – parametr *-l* nastavuje jmenovku svazku, která je pak důležitá pro nastavení v clusteru:

```
mkqdisk -c /dev/sdc1 -l proxmox_qdisk
```

10. Na druhém z uzlů, na kterém jsme formátování quorum disku neprováděli, nejprve odpojíme iscsi disk a opětovně připojíme:

```
iscsiadm -m node --targetname "iqn.2016-02.lan.test.quorum
   :qdisk" --portal "160.217.161.169:3260" --logout &&
iscsiadm -m node --targetname "iqn.2016-02.lan.test.quorum
   :qdisk" --portal "160.217.161.169:3260" --login
```

Quorum disk je připravený a nyní již dojde k poslední fázi konfigurace a to k nastavení Proxmox jako HA clusteru.

11. Na jednom z uzlů zkopírujeme soubor obsahující informace pro HA cluster:

```
cp /etc/pve/cluster.conf /etc/pve/cluster.conf.new
```

12. A upravíme nově vytvořený soubor */etc/pve/cluster.conf.new*, aby vypadal následovně:

```
<?xml version="1.0"?>
<cluster name="cluster-test" config_version="3">

 <cman expected_votes="3" keyfile="/var/lib/pve-cluster/
    corosync.authkey"/>
 <quorumd votes="1" allow_kill="0" interval="1" label="
    proxmox_qdisk" tko="10"/>
 <totem token="54000"/>

 <fencedevices>
  <fencedevice agent="fence_apc" ipaddr="192.168.1.101"
     login="apc" name="apc" passwd="apc" power_wait="10"
     />
 </fencedevices>

 <clusternodes>
  <clusternode name="node1" votes="1" nodeid="1">
```

```
    <fence>
      <method name="power">
        <device name="apc" port="1" secure="on"/>
      </method>
    </fence>
   </clusternode>
      <clusternode name="node2" votes="1" nodeid="2">
     <fence>
      <method name="power">
        <device name="apc" port="2" secure="on"/>
      </method>
     </fence>
    </clusternode>
  </clusternodes>
</cluster>
```

Číslo u položky na druhém řádku *config_version* se zvyšuje o jedna oproti aktuální verzi.

13. Ověříme, že je nový konfigurační soubor korektní:

```
ccs_config_validate -v -f /etc/pve/cluster.conf.new
```

14. Přes webové rozhraní (obrázek 4.15) v levé části vybereme položku *Datacenter* a v hlavním okně v záložce *HA* klikneme na *Activate*. Tím se na cluster aplikují změny provedené v souboru *cluster.conf.new* a dojde k jeho distribuci na ostatní uzly.

15. Na obou uzlech upravíme soubor */etc/default/redhat-cluster-pve* tak, že odkomentujeme řádku:

```
FENCE_JOIN="yes"
```

16. Pak na obou uzlech provedeme následující příkazy (každou řádku samostatně):

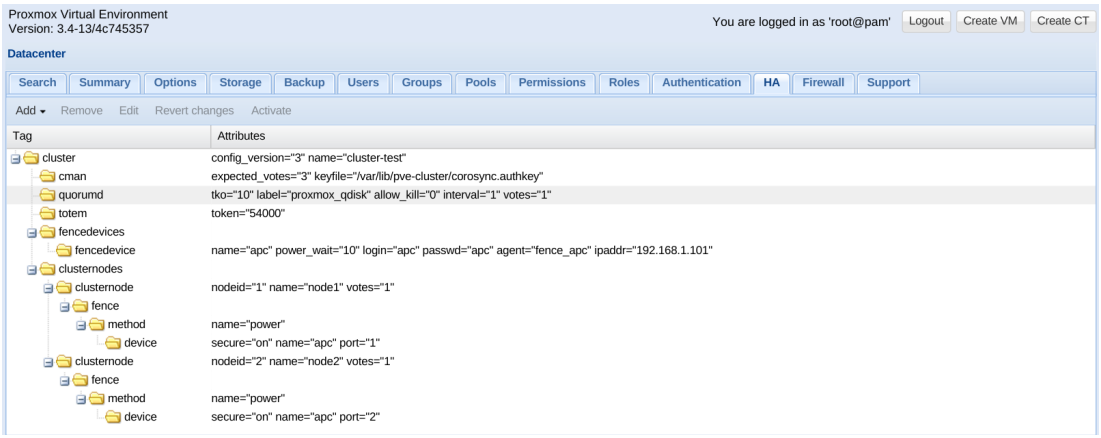
```
/etc/init.d/rgmanager stop
/etc/init.d/cman reload
/etc/init.d/rgmanager start
/etc/init.d/pve-cluster restart
```

Tím se nová konfigurace aplikuje.

17. Nakonec ověříme stav clusteru a jeho uzlů pomocí příkazu:

```
clustat
```

Pokud jsou všechny uzly online a cluster je usnášeníschopný, je vše hotovo.



Obrázek 4.15: Proxmox – finální konfigurace HA clusteru (autor)

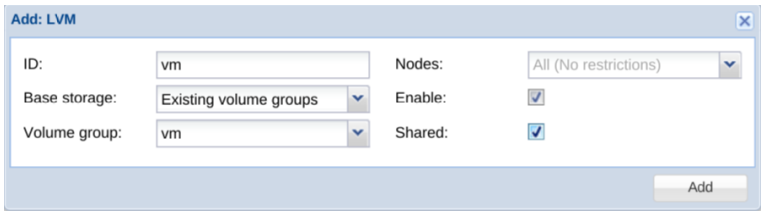
4.6 Virtuální stroje

Vysoce dostupný cluster je připravený k provozu, a proto vyzkoušíme nastavit prostředí pro spouštění virtuálních počítačů. Nejprve upravíme základní parametry clusteru. Nastavíme si, kde bude úložiště instalačních médií pro jednotlivé virtuální stroje a určíme, kde budou uloženy jejich virtuální pevné disky. Vše již provádíme pohodlně přes webové prostředí (obrázek 4.17).

1. Ve webovém rozhraní v levé části vybereme položku *Datacenter* a v hlavním okně v záložce *Storage*. Vybereme položku *local* a vybereme *Edit*.

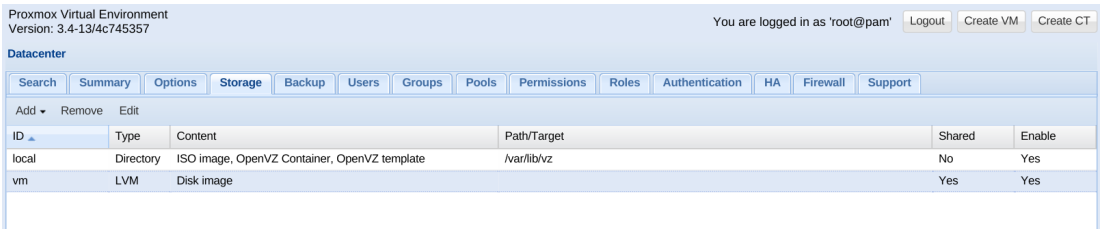
2. V položce *Content* odebereme *Disk image*, protože budeme využívat jiný systém. Zde budeme ukládat pouze binární kopie (ISO) instalačních médií jednotlivých operačních systémů.

3. Přidáme nové úložiště *Add – LVM*, *ID* nazveme *vm*, *Volume group* vybereme *vm* a zaškrtneme *Shared*:



Obrázek 4.16: Proxmox – nastavení LVM úložiště (autor)

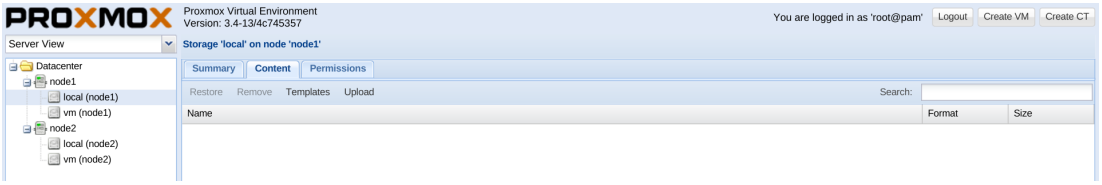
Zaškrtnutí položky *Shared* je velice důležité pro chod v clusteru. Systém si tak hlídá jednotlivé LVM svazky, aby nebyl jeden svazek používán současně na obou uzlech.



Obrázek 4.17: Proxmox – finální konfigurace úložiště clusteru (autor)

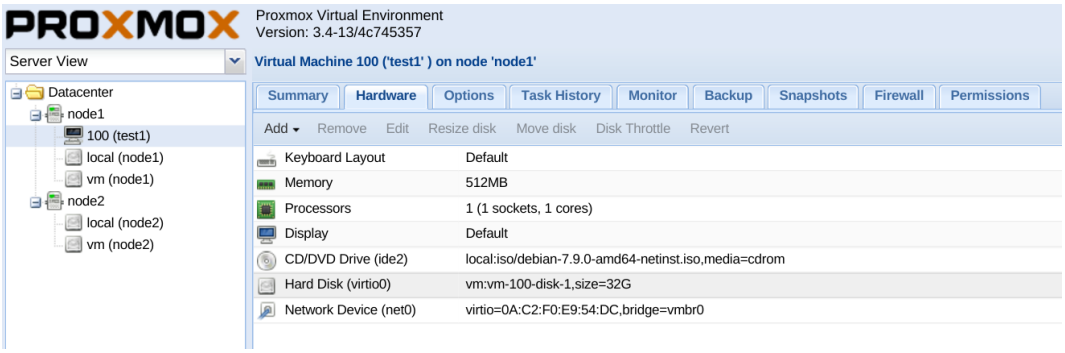
Nyní můžeme nahrát první instalační ISO do našeho adresáře. Vždy je dobré nahrávat tento soubor na všechny uzly, protože složka s ISO image není synchronizována mezi jednotlivými uzly.

Ve webovém rozhraní (obrázek 4.18) v levé části vybereme položku *local (node1)* a v hlavním okně v záložku *Content*. Klikneme na *Upload* a vybereme soubor k nahrání. To samé opakujeme i u druhého uzlu, tj. položky *local (node2)*.



Obrázek 4.18: Proxmox – nahrání instalačního ISO obrazu (autor)

Zbývá pouze vytvořit prostředí pro virtuální stroj, což provedeme ve webovém rozhraní kliknutím vpravo nahoře na *Create VM* a vyplníme jaké parametry chceme dislokovat pro virtuální stroj (jméno, typ OS, instalační ISO médium, velikost pevného disku, procesor, paměť, sítová karta atd.). Podrobnosti VM stroje jsou k dispozici na záložce *Hardware* (obrázek 4.19).



Obrázek 4.19: Proxmox – konfigurace virtuálního stroje (autor)

Samotné spuštění virtuálního stroje se provádí tlačítkem *Start*, které je vpravo nahoře, nebo je možné otevřít virtuální obrazovku tlačítkem *Console* a pak po kliknutí na ikonu ON/OFF vybrat start stroje (obrázek 4.20).

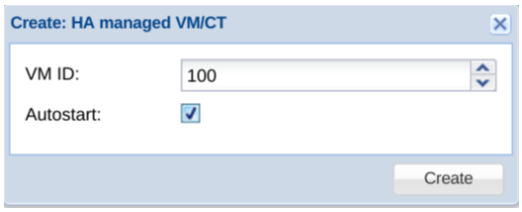
Po nainstalování operačního systému je nutné odpojit ISO, které je navázané na CD-ROM mechaniku virtuálního stroje, jinak není možné tento stroj migrovat mezi uzly.

Obrázek 4.20: Proxmox – obrazovka virtuálního stroje (autor)

Nyní si nově vytvořený virtuální stroj přidáme mezi služby udržované HA clusterem.

1. Přes webové rozhraní v levé části vybereme položku *Datacenter* a v hlavním okně v záložce *HA* klikneme na *Add* a vybereme *HA managed VM/CT*.

2. Do položky *VM ID* napíšeme identifikační číslo našeho stroje, klikneme *Create*:

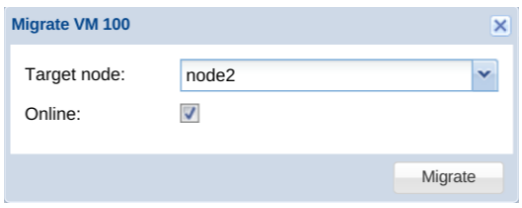


Obrázek 4.21: Proxmox – nastavení VM jako HA (autor)

3. Pro aktivaci nové konfigurace klikneme v menu HA záložky na *Activate*.

Pokud by nyní došlo k nějakému technickému problému, například výpadku prvního uzlu, na kterém v dané chvíli virtuální stroj poběží, HA cluster to okamžitě pozná a spustí tento stroj na druhém uzlu. Výpadek se tak minimalizuje na řády sekund.

Migraci mezi jednotlivými uzly lze také provádět manuálně, například z důvodu údržby apod. Proxmox podporuje i moderní a užitečnou migraci běžícího virtuálního stroje na jiný uzel s nulovým výpadkem (tzv. živá migrace). Manuální migrace se provádí pomocí tlačítka *Migrate* v pravé části webového rozhraní (obrázek 4.22).

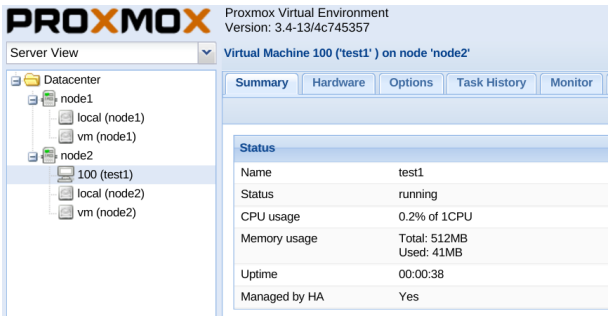


Obrázek 4.22: Proxmox – migrace virtuálního stroje na jiný uzel (autor)

4.7 Simulace výpadku jednoho z uzlů

Abychom vyzkoušeli správnou funkci našeho systému, podrobíme jej simulaci výpadku jednoho z uzlů. Provedeme to tak, že zcela nečekaně (bráno z pohledu systému) odpojíme tento uzel od přívodu elektrické energie. Na postiženém uzlu bude v danou chvíli spuštěný virtuální stroj, který budeme monitorovat pomocí příkazu *ping* a budeme sledovat dobu, po kterou se stane tento stroj nedostupný.

1. Vše je online:



Obrázek 4.23: Proxmox – stav před simulovaným výpadkem (autor)

2. Kabel napájení je fyzicky odpojen.

3. Do 3 sekund síťové přepínače a bond zařízení zjistily, že není funkční jedna z linek LACP a veškerý provoz přesměrovaly pouze přes funkční linky.

4. Po 12 sekundách systém detekoval, že *node2* je nedostupný.

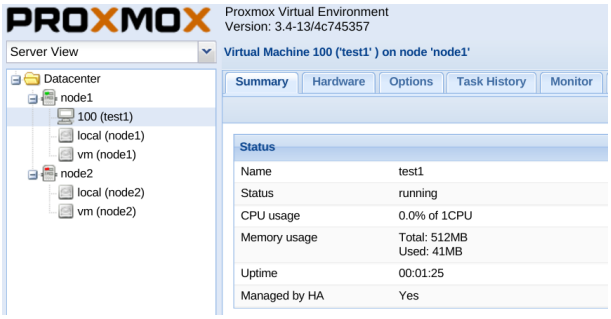
5. V souboru *cluster.conf* máme definován *totem token* na 54 000 mikrosekund. To znamená, že doba od zjištění nedostupnosti uzlu do prohlášení, že uzel je skutečně nedostupný je 54 sekund. Cluster po tuto dobu čeká, zda se mu uzel *node2* přihlásí.

6. Po 66 sekundách od odpojení kabelu se uzel *node2* stále neohlásil a tak se cluster snaží zajistit bezpečnou migraci na funkční uzel tak, aby nedošlo ke spuštění stejného virtuálního stroje na obou uzlech současně. Protože *node2* nekomunikuje, je nucen použít STONITH příkaz a nefunkční uzel odpojit od přívodu elektrické energie. Tím má cluster zaručeno, že na nekomunikujícím uzlu není spuštěný žádný virtuální stroj a může tak dát k dispozici poskytování tohoto virtuálního stroje jinému funkčnímu uzlu.

7. Po použití fencingu (STONITH) máme nastavenou desetisekundovou prodlevu, aby měl systém napájení dostatek času k vypnutí přívodu elektrické energie (parametr *power_wait* u *fencedevice* v souboru *cluster.conf*).

8. Po 76 sekundách je virtuální stroj automaticky přesunutý na uzel *node1* (obrázek 4.24).

9. Po 92 sekundách je virtuální server plně funkční, včetně jeho operačního systému.



Obrázek 4.24: Proxmox – virtuální stroj automaticky přenesen na nový uzel (autor)

Vše proběhlo zcela automaticky, bez jakéhokoliv manuálního zásahu administrátora. Doba reakce systému na neočekávanou událost je velice rychlá, zvláště vezmeme-li v potaz, že za dobu 92 sekund nemusí administrátor ani zjistit nedostupnost uzlu či virtuálního stroje, natož vzniklý problém vyřešit.

Doba čekání na to, než je uzel skutečně prohlášen za nedostupný (54 sekund) je doporučena výchozí instalací, ale je možné tuto hodnotu upravit. Běžně se používá doba 30 sekund i méně. Vždy je dobré otestovat velikost čekací doby na konkrétním řešení a daném připojení quorum disku. Pokud by byla hodnota příliš nízká, mohlo by zbytečně docházet k prohlašování nedostupnosti uzlu i v případech, kdy by došlo k velice mimořádnému a krátkodobému výpadku, nebo by na výpadek nestihly zareagovat ostatní ochranné prvky.

4.8 Náklady

V příloze B na straně 44 je v tabulce zpracováno cenové porovnání běžného a clusterového řešení. HA cluster vychází více jak dvakrát dražší. Hlavním důvodem je nutná duplikace veškerého vybavení a síťového připojení, která se týká převážně prvotní investice. Celkovou částku ovlivňuje zejména výrazně vyšší pořizovací cena síťových přepínačů, protože musí podporovat vzájemnou komunikaci (třetí a čtvrtá řádka tabulky).

Provozní náklady se obvykle maximálně zdvojnásobí, nebo dokonce úměrně klesnou díky množstevní slevě poskytovatele hostingu a efektivnějšímu využívání práce administrátora s ohledem na identickou konfiguraci systému.

Z tabulky je zřejmé, že vysoce dostupné řešení v porovnání s běžným je poměrně nákladnou záležitostí. Pro mnoho firem a organizací je dostupnost jejich informačních služeb životně důležitá. Škody vzniklé případným výpadkem informačního systému by byly výrazně vyšší než pořizovací cena, a proto ve svých rozpočtech počítají s pořízením (případně pronájmem) a provozem HA clusteru.

5. Závěr

Podstatná část této práce teoreticky popisuje základní prvky důležité pro implementaci vysoce dostupného clusteru včetně technologií, které jsou nutné k jeho realizaci. Bez základních teoretických znalostí nelze implementaci úspěšně provést, proto byl této části věnován dostatečný prostor.

Zbývající část je pak praktickým návodem implementace vysoce dostupného clusteru pomocí operačního systému Linux a nadstavby Proxmox. Postupně je vše detailně popsáno od návrhu rozložení jednotlivých komponent, přes instalaci základního operačního systému až po nastavení systému jako HA clusteru. Následně je ukázáno využití clusteru pro poskytování virtuálního prostředí. Jako důkaz úspěšné implementace je nasimulován nečekaný výpadek jednoho z uzlů clusteru a zdokumentováno jak systém na danou situaci reaguje.

Tato práce může posloužit jako návod ostatním IT administrátorům, jak realizovat vlastní vysoce dostupné řešení prostřednictvím otevřeného software (opensource). Také může pomoci firmám nebo organizacím při rozhodování o nasazení podobného systému v jejich firemním prostředí.

Jak bylo již zmíněno výše, hlavní nevýhodou HA clusteru je jeho finanční náročnost z důvodu nutné redundance všech jeho prvků. Duplikace všech částí systému samozřejmě zvyšuje i provozní náklady, jako je spotřeba elektrické energie nebo požadavky na schopnosti systémového administrátora. Proti tomu stojí nesporná výhoda schopnosti zaručit vysokou dostupnost IT služeb poskytovaných clusterem. Vždy je nutné posoudit vhodnost a efektivnost tohoto řešení individuálně, konkrétně pro daný případ. Je důležité provést hlubší analýzu požadavků na systém a celkovou rentabilitu investice vložené do systému.

Z důvodu osobní několikaleté zkušenosti a popsaných schopností vysoce dostupného řešení docházím k závěru, že v případě, kdy je určitá organizace nebo firma výrazně závislá na své IT infrastruktuře, měla by zanalyzovat a posoudit nasazení HA clusteru.

Summary

This bachelor thesis is dedicated to the part of IT (information technology) field focusing on technical background and providing of services. It describes how to create high availability computer systems and how to protect them from unexpected failures. Many state and private companies know that the right implementation is very important for provisioning of IT services. The thesis compares the early investment cost against the gained benefits.

The objective of this work is to describe in detail the process of creating and implementation of the HA cluster based on operating system Linux with Proxmox extension. The demands on hardware and software equipment are gradually presented here. The physical connections and configurations of individual parts of the system are analyzed to find the optimal design. Several options of virtualization platforms which are available on the market are listed here. Demands on the financial investment and consequent implementation of the HA cluster depend on the selected type of solution.

Keywords: IT (information technology); cluster; high availability; providing of services; linux; virtualization; computer systems;

Literatura

Cheng, S. M. (2014). *Proxmox high availability*. Packt Publishing Ltd. Birmingham, UK.

Cisco, I. (2016). *Data Center High Availability Clusters.* [Online]. Dostupné 20/2/2016, z http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Data_Center/HA_Clusters/HA_Clusters/HAOver_1.pdf

Hellman, B., Haas, F., Reisner, P., & Ellenberg, L. (2012). *DRBD Users Guide 8.4.* [Online]. Dostupné 10/2/2016, z http://www.drbd.org/en/doc/users-guide-84

Oracle. (2016). *Fusion Middleware High Availability Guide.* [Online]. Dostupné 14/2/2016, z http://docs.oracle.com/cd/E17904_01/core.1111/e10106/toc.htm

Proxmox, S. S. G. (2016). *Proxmox VE documentation.* [Online]. Dostupné 16/2/2016, z https://pve.proxmox.com/wiki/Main_Page

RedHat, I. (2016). *RedHat Enterprise Linux Cluster High Availability, and GFS Deployment Recommended Practices.* [Online]. Dostupné 1/2/2016, z https://access.redhat.com/articles/40051

van Vugt, S. (2014). *Pro linux high availability clustering*. Apress, CA.

Zdroje obrázků

Administrator's Guide for Release 6. (2016). [Obrázek]. Dostupné 14/2/2016, z https://docs.oracle.com/cd/E37670_01/E41138/html/ch18s07.html

Austin, S. (2012). *Server Virtualization Part 1 – A Crash Course on an IT Game Changer*. [Obrázek]. Dostupné 16/2/2016, z http://www.animate.com/server-virtualization-part-1-a-crash-course-on-an-it-game-changer-2/

Hammond, J. (2007). *Link aggregation*. [Obrázek]. Dostupné 14/2/2016, z https://en.wikipedia.org/wiki/Link_aggregation

Herbert, G. W. (2006). *High-availability cluster*. [Obrázek]. Dostupné 11/2/2016, z https://en.wikipedia.org/wiki/High-availability_cluster

Jones, M. T. (2010). *High availability with the Distributed Replicated Block Device*. [Obrázek]. Dostupné 16/2/2016, z http://www.ibm.com/developerworks/library/l-drbd/

Layton, J. B. (2010). *Storage Management with an LVM GUI*. [Obrázek]. Dostupné 14/2/2016, z http://www.linux-mag.com/id/7796/

Leicht, D. (2007). *Cluster Scheme*. [Obrázek]. Dostupné 10/2/2016, z https://en.wikipedia.org/wiki/File:Cluster_Scheme_New.JPG

Rich, B. (2011). *Oracle Fusion Middleware*. [Obrázek]. Dostupné 10/2/2016, z http://docs.oracle.com/cd/E17904_01/core.1111/e10106/intro.htm#ASHIA713

Seznam obrázků

Rejstřík

Přílohy

A. Doporučená konfigurace

Doporučená konfigurace pro realizaci specifického projektu včetně porovnání řešení pomocí HA clusteru nebo standardní jeden server.

Zadávací požadavek:

- vlastní server pro virtualizované prostředí
- maximálně 10 virtuálních strojů
- obvyklá konfigurace VM: 2-4 × vCPU, 4-8 GB RAM, HDD 200-300 GB
- maximální dostupnost všech strojů
- umístnění v datovém centru v Praze (není nutné řešit záložní napájení)
- přímý uplink do internetu 1 Gb/s
- předpokládaná životnost 5 let

Byly vybrány následující komponenty systému:

- Server Supermicro:
  - procesor Intel Xeon E5-1620v2 3,7GHz 10 MB cache 4core HT
  - základní deska Supermicro X9SRL-F, integrováno 2 × 1 Gb/s port LAN
  - IPMI modul pro vzdálenou správu (rovněž nutné pro fencing)
  - paměť RAM 64 GB DDR3 ECC Registered
  - 2 × pevný disk Western Digital 3 TB Raid Edition 7200 rpm
  - redundatní napájecí zdroj
  - provedení 1U
- Přepínače pro HA cluster: Netgear GS728XTS (zapojené do stacku)
- Přepínače pro standardní server: Netgear GS108T
- Pro HA cluster je nunté přidat: 2 × 1 Gb/s port LAN karta (do každého serveru)

B. Cenové porovnání

Řešení		**HA cluster**		**Standardní**	
Komponenta	*cena za kus*	*počet*	*celkem*	*počet*	*celkem*
Server pro HA	75 000	2	150 000	–	–
Server standard	70 000	–	–	1	70 000
Switch pro HA	17 000	2	34 000	–	–
Switch standard	2 500	–	–	1	2 500
Celková cena za HW			**184 000**		**72 500**
Provozní položka	*cena za měsíc*	*měsíců*	*celkem*	*měsíců*	*celkem*
Hosting	1 800	120*	216 000	60	108 000
Administrace	3 000	120*	360 000	60	180 000
Celková provozní cena			**576 000**		**288 000**
Celková cena			**760 000 Kč**		**360 500 Kč**

* *dva servery za 5 let (2 servery × 60 měsíců)*

Celkovou cenu za hardware lze chápat jako pořizovací cenu, tedy jednorázový výdaj, který je nutný na pořízení systému. Celková provozní cena vyjadřuje předpokládané pravidelné měsíční výdaje po dobu pěti let nutné k údržbě a chodu systému.