

Konfigurace failover klastru

Vojtěch Novotný

Bakalářská práce
2022



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Vojtěch Novotný
Osobní číslo:	A18095
Studijní program:	B3902 Inženýrská informatika
Studijní obor:	Inteligentní systémy s roboty
Forma studia:	Prezenční
Téma práce:	Konfigurace failover klastru
Téma práce anglicky:	Configuring Failover Cluster

Zásady pro vypracování

1. Vytvořte literární rešerši na téma failover cluster.
2. Seznamte se s klíčovými pojmy a technikami nutnými k instalaci.
3. Zaměřte se na správu dlouhodobě běžících strojů.
4. Ověřte funkčnost a odolnost systému proti poruchám experimentálními testy.
5. Věnujte pozornost zabezpečení.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Red Hat. Red Hat Enterprise Linux. 2021, dostupné z: <https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux>
2. Red Hat. Product Documentation for Red Hat Enterprise Linux 8. 2021, dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8
3. Comunity. GitHub – ClusterLabs/crmlsh: Command-line interface for High-Availability cluster management on GNU/Linux systems. 2021, dostupné z: <https://github.com/ClusterLabs/crmlsh>
4. HIGH AVAILABILITY CLUSTER: Technology White Paper. EuroNAS [online]. 2018 [cit. 2021-11-13]. Dostupné z: https://euronas.com/wp-content/uploads/2018/08/euroNAS_HA_Cluster_White_Paper.pdf
5. Configuration and Administration Basics. SUSE [online]. [cit. 2021-11-13]. Dostupné z: <https://documentation.suse.com/sle-ha/15-SP1/html/SLE-HA-all/cha-ha-config-basics.html>
6. Red Hat. Basic and advanced configuration of Security-Enhanced Linux (SELinux). 2021, dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/using_selinux/index
7. Red Hat. Using nftables in Red Hat Enterprise Linux 8. 2019, dostupné z: <https://www.redhat.com/en/blog/using-nftables-red-hat-enterprise-linux-8>

Vedoucí bakalářské práce:

doc. Ing. Martin Sysel, Ph.D.
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **15. ledna 2022**

Termín odevzdání bakalářské práce: **20. května 2022**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan



prof. Ing. Vladimír Vašek, CSc. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo - bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnaní případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 20.5.2022

Vojtěch Novotný, v. r.
.....
podpis studenta

ABSTRAKT

Bakalářská práce se zabývá problematikou vysoce dostupných klastrů na platformě Linux. Jsou popsány informace z oblasti použití klastrů a jejich provozu. Teoretická část popisuje počítačové klastry, typy a význam jejich použití. V úvodu praktické části práce je popsán návrh failover klastru tvořený nástroji Pacemaker a Corosync na Red Hat forku a struktura těchto nástrojů. Obsažen je také popis linuxových distribucí a konkrétněji vysvětlena Red Hat virtualizace.

Ve čtvrté kapitole je popsán návrh, instalace a konfigurace jednotlivých uzlů klastru a serveru, který poskytuje klastru sdílené úložiště. Poslední část čtvrté kapitoly se věnuje experimentálním testům, které mají za cíl ověřit navržený failover klaster a otestovat dostupnost služeb, které klaster spravuje. Jako služba byl spuštěn webserver napojený na databázi, jako nejčastější příklad implementace na failover klastru. Poslední kapitola se zabývá možným rozšířením této práce například zabezpečením celého řešení.

Klíčová slova: klaster, failover, Red Hat, Pacemaker, Corosync

ABSTRACT

The bachelor thesis deals with issue of highly available clusters on the Linux platform. Thesis describes informations about the use of cluster and their service. The theoretical part describes computer clusters, types, and their use. The introduction of the practical part describes the design of a failover cluster consisting of Pacemaker and Corosync tools on the Red Hat fork and the structure of these tools. A description of Linux distributions and more specifically Red Hat virtualization, is also included.

The fourth chapter describes the design, installation and configuration of individual cluster nodes and the server that provides shared storage to the cluster. The last part of the fourth chapter deals with experimental tests, which aim to verify the proposed failover cluster and test the availability of services that the cluster manages. A webserver connected to the database was launched as a service, as the most common example of a failover cluster implementation. The last chapter deals with possible extensions of this work, such as securing the whole solution.

Keywords: cluster, failover, Red Hat, Pacemaker, Corosync

Rád bych poděkoval vedoucímu bakalářské práce panu Doc. Ing. Martinu Syslovi, Ph.D. za jeho odborné vedení, cenné připomínky a za jeho čas při řešení dané problematiky.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 KLASTR.....	11
1.1 HISTORIE.....	11
1.2 POJEM KLASTR	11
1.3 TYPY KLASTRŮ.....	12
1.3.1 Vysoce výkonné klastry (High performance - HP).....	12
1.3.2 Klastry pro vyrovnávání zátěže (Load balancing - LB).....	12
1.3.2.1 Network-based vyrovnávání zátěže	12
1.3.2.2 Hardware-based vyrovnávání zátěže	12
1.3.3 Klastry s vysokou dostupností (High availability - HA).....	12
1.3.3.1 Převzetí služeb při selhání (failover schopnost)	13
1.4 KLASIFIKACE KLASTRU	13
1.4.1 Komponenty klastrového počítače	14
1.5 MOTIVACE KLASTROVÉHO SYSTÉMU.....	14
1.6 HROZBY OHROŽUJÍCÍ PROVOZ APLIKACE	14
1.7 ŘEŠENÍ HROZEB.....	16
1.7.1 Geograficky distribuovaný klastr	16
1.7.2 Výhody klastrového počítače	17
1.7.3 Nevýhody klastrového počítače	18
2 FAILOVER KLASTR	19
2.1 SDÍLENÉ ÚLOŽIŠTĚ	19
2.1.1 Klastrové souborové systémy	19
2.2 SOUBOROVÉ SYSTÉMY NA RED HAT.....	20
2.2.1 Typy souborových systémů.....	20
2.2.2 Přehled lokálních souborových systémů	20
2.2.2.1 Souborový systém XFS	21
2.2.2.2 Souborový systém Ext4	22
2.2.3 Síťové souborové systémy	23
2.2.4 Souborové systémy sdíleného úložiště.....	24
2.3 PACEMAKER.....	25
2.3.1 Corosync	26
2.4 ARCHITEKTURA PACEMAKERU.....	26
2.4.1 Součásti Pacemakeru.....	26
2.4.2 Klastrové prostředky	28
2.4.3 Třídy prostředků	28
2.4.4 Korum	29
2.4.5 Oplocení [fencing]	30
II PRAKTICKÁ ČÁST	31
3 LINUX.....	32

3.1	SLACKWARE.....	33
3.2	DEBIAN	34
3.3	RED HAT	34
3.3.1	RHEL	35
3.3.1.1	RHEL 8	35
3.4	RED HAT DISTRIBUCE.....	37
3.4.1	CentOS Stream 8.....	37
3.4.2	AlmaLinux	38
3.4.3	Rocky Linux 8.....	39
3.4.4	Souhrn z distribucí	39
3.5	VIRTUALIZACE	39
3.5.1	Typy virtualizace.....	39
3.5.2	Typ 1 (nativní, plný)	40
3.5.3	Typ 2 (hostovaný)	41
4	STUDENÝ POHOTOVOSTNÍ KLASTR.....	42
4.1	INSTALACE A KONFIGURACE SERVERU.....	43
4.1.1	Sdílené úložiště	43
4.1.1.1	Konfigurace iSCSI klientů.....	46
4.1.2	Použití HA-LVM	47
4.2	DRBD.....	48
4.2.1	Nástroje pro správu	49
4.2.1.1	Drbdadm	49
4.2.1.2	Drbdsetup.....	49
4.2.1.3	Drbdmeta	50
4.2.2	Role zdrojů	50
4.3	INSTALACE A KONFIGURACE UZLŮ KLASTRU	51
4.3.1	Konfigurace HA-LVM v klastru	53
4.3.2	Konfigurace Filesystem v klastru.....	54
4.3.3	Aplikace	55
4.3.3.1	Server Apache	55
4.3.3.2	PHP	55
4.3.3.3	Databáze MariaDB	56
4.3.3.4	Instalace aplikace	56
4.3.3.5	Přidání prostředků aplikace do klastru.....	58
4.4	KONFIGURACE KLASTRU NA WEBU	59
4.5	EXPERIMENTÁLNÍ TESTY	60
4.5.1	Výpadek při odstavení klastru na uzlu	61
4.5.2	Výpadek při tvrdém vypnutí uzlu	63
5	ZABEZPEČENÍ	65
5.1	DŮLEŽITÉ ZABEZPEČENÍ.....	65
5.1.1	Dobře nastavená politika hesel.....	65
5.1.2	Logické oddělení sítí	65
5.1.3	Vygenerování SSH klíčů.....	66
5.2	VYŠŠÍ STUPEŇ ZABEZPEČENÍ	67
5.2.1	Zakázat přihlášení pod uživatelem root	67

5.2.2	Aktualizace.....	67
5.2.3	Firewall	67
5.3	NEJVYŠŠÍ STUPEŇ ZABEZPEČENÍ.....	68
5.3.1	Nftables	68
5.3.2	SELinux.....	68
ZÁVĚR		70
SEZNAM POUŽITÉ LITERATURY.....		71
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....		77
SEZNAM OBRÁZKŮ		79
SEZNAM TABULEK.....		81
SEZNAM PŘÍLOH.....		82

ÚVOD

V posledních letech zaznamenala poptávka po úložištích s vysokou dostupností prudký nárůst. Společnosti jsou stále více závislé na obsahu datového úložiště. Spolehlivost a dostupnost datových úložišť bylo třeba dramaticky zvýšit, aby se předešlo vážným následkům, které může mít selhání serveru. K dosažení co nejvyšší dostupnosti a co nejrychlejší obnově po katastrofální nehodě jsou vyžadovány úložné systémy, které dokáží tuto krizi řešit. Více než kdy jindy je důležité zachovat obchodní kontinuitu i při úplném selhání serveru. Vysoce dostupný klastr se používá k poskytování důležitých sdílených dat v síti, jako například systémové a datové disky pro fyzické a virtuální produkční servery a produkční pracovní stanice. Při provozu virtualizovaného prostředí nebo při plnění požadavku 24 hodinového obchodního cyklu, si nelze dovolit prostoje [1].

Při použití vysoce dostupného klastru budou síťové aplikace nadále fungovat, bez ohledu na to, který server by mohl selhat. Klienti budou mít stále přístup ke svým datům. Standardní servery NAS mohou poskytovat mnoho redundantních funkcí, jako je například RAID, selhání síťového portu atd., ale nechrání před úplným selháním serveru. I se správnou politikou zálohování chvíli potrvá a vyžádá si manuální zásah, než se server a data vrátí zpět do online stavu. U vysoce dostupného klastru, pokud dojde k hardwarové nebo softwarové chybě, bude-li tato chyba detekována pomocí inteligentních funkcí, software automaticky a téměř okamžitě přesune požadavky na jiný server zrcadlený v klastru [1][2].

Znalosti čerpány z teoretické části této práce budou otestovány experimentálními testy v části praktické. V praktické části bude v klastru vytvořeném na Linuxové distribuci nasazen webový server. Webová stránka bude napsána v jazyce PHP a bude napojena na MariaDB databázi, která bude zaznamenávat uživatele, kteří podají komentář. Odolnost webového serveru bude testována vypínáním aktivního uzlu, pokusem na obou uzlech provést údržbu, například spustit update, či přidat virtuálně RAM paměť a bude zdokumentována případná nedostupnost aplikace v rámci těchto akcí.

I. TEORETICKÁ ČÁST

1 KLASTR

1.1 Historie

První inspirace pro klastrové výpočty byla vyvinuta v 60. letech 20. století společností IBM jako alternativa propojení velkých sálových počítačů, za účelem poskytnutí nákladově efektivnější formy výpočetních jednotek. V té době systém IBM Houston Automatic Spooling Priority a jeho nástupce Job Entry System umožňovali distribuci práce do uživatelem vytvořeného klastru sálových počítačů. IBM stále podporuje klastrování sálových počítačů, prostřednictvím jejich systému Parallel Sysplex, který umožňuje hardwaru, operačnímu systému, middlewaru a softwaru pro správu systému poskytovat dramatické zvýšení výkonu při nižších nákladech, a zároveň umožňuje velkým uživatelům sálových počítačů nadále spouštět jejich stávající aplikace [3]. Klastrování nabralo na síle a rozmachu až s konvergencí tří důležitých trendů v 80. letech – vysoce výkonných mikroprocesorů, vysokorychlostních sítí a standardizovaných nástrojů pro vysoce výkonné distribuované výpočty. Možným čtvrtým trendem je rostoucí potřeba výpočetního výkonu pro výpočetní vědu a komerční aplikace spojená s vysokou cenou a nízkou dostupností tradičních superpočítačů. Pokroky v těchto technologiích a jejich stále větší dostupnost jako levných a komoditních komponent činí z klastrů nebo sítí počítačů, jako jsou osobní počítače, pracovní stanice a symetrické vícenásobné procesory (multiprocessing), atraktivní řešení pro nákladově efektivní paralelní výpočty [4].

Vznik klastrových platforem byl řízen řadou akademických projektů, jako je Beowulf, Berkeley NOW a HPVM, které dokazují výhodu klastrů oproti jiným tradičním platformám. Tyto výhody zahrnují nízké vstupní náklady na přístup k výkonu na úrovni superpočítačů, možnost sledování nových trendů a jejich adaptabilita, postupně upgradovatelné systémy, open source vývojové platformy a nezávislost na jednom prodejci. Klastry jsou dnes široce využívány pro výzkum a vývoj vědy, techniky, komerční a průmyslové aplikace, které vyžadují vysoce výkonné výpočty. Navíc klastry nabízí i výhody, jako je vysoká dostupnost a škálovatelnost, které motivují k širokému použití i v ne-superpočítačových aplikacích, jako jsou klastry fungující jako webové a databázové servery [4].

1.2 Pojem klastr

Klastrování je soubor těsně nebo volně propojených počítačů, které vykazují velmi vysoký výkon, ale z pohledu koncových uživatelů (zákazníků), se klastr jeví jako samostatný

system. Propojené počítače provádějí operace podle našeho předem definovaného plánu a vytváří tak myšlenku silného systému. Klastry jsou obecně propojeny prostřednictvím rychlých lokálních sítí (LAN).

1.3 Typy klastrů

1.3.1 Vysoce výkonné klastry (High performance - HP)

Vysoce výkonné klastry využívají počítačové klastry a superpočítače k řešení pokročilých výpočetních problémů. Jsou zvyklé vykonávat funkce, při kterých je potřeba vzájemná komunikace mezi uzly, při provádění svých úloh. Jsou navrženy tak, aby využívali výhody paralelního výpočetního výkonu několika uzlů [5].

1.3.2 Klastry pro vyrovnávání zátěže (Load balancing - LB)

1.3.2.1 *Network-based vyrovnávání zátěže*

Při stále větším využití sítě a využívání internetu představuje vyrovnávání zátěže klíčový faktor. V těchto klastrech zůstávají všechny uzly integrovány do všech instancí, takže všechny tyto entity uzlů jsou si vědomy požadavků ve své síti. Systémy nepracují společně v jednom postupu, ale požadavky na uzly jsou adresovány separátně, na základě algoritmu plánovače. Plánovač pomáhá zefektivňovat síť, distribuuje zpracování a provoz rovnoměrně po síti a zajišťuje, že žádné jeho zařízení není zahlceno [6].

1.3.2.2 *Hardware-based vyrovnávání zátěže*

Pokud klastř s povoleným vyrovnáváním zatížení obdrží požadavek od jedné ze služeb, náhodně vybere člena klastru. Pokud je tento člen ovšem na limitu nebo přes limit omezení nastavený pro CPU nebo paměť, je vybrán jiný uzel v klastru. Pokud jsou všichni členové v klastru ve stejném stavu, požadavek se nezdaří [6].

1.3.3 Klastry s vysokou dostupností (High availability - HA)

Klastry s vysokou dostupností jsou navrženy tak, aby udržovaly synchronní redundantní uzly, které mohou fungovat jako záložní systémy v případě, že dojde k selhání. Čím větší je počet počítačů zahrnutých v klastru, tím menší je pravděpodobnost selhání poskytované služby. Počítače, které jsou součástí geograficky rozptýleného klastru, také poskytují navíc ochranu před přírodními katastrofami, teroristickými útoky a dalšími hrozbami. Jsou navrženy a poskytovány tam, kde jsou konzistentní počítačové služby, jako jsou obchodní

aktivity, složité databáze, zákaznické služby, elektronické webové stránky a síťová distribuce souborů. Jejich cílem je zákazníkovi poskytovat nepřetržitou dostupnost dat [7].

1.3.3.1 Převzetí služeb při selhání (failover schopnost)

Vysoká dostupnost je definována, jako schopnost systému pokračovat ve fungování po selhání jednoho nebo více serverů. Součástí vysoké dostupnosti je převzetí služeb při selhání, které je definováno jako schopnost klientských připojení migrovat z jednoho serveru na druhý a v případě selhání serveru mohou klientské aplikace pokračovat v provozu [5].

Nejobecnější dělení failover klastrů podle tří základních principů:

- Studené pohotovostní klastry

V tomto případě aktivní uzel zpracovává požadavky a pasivní uzly jsou nečinné a čekají, až aktivní uzel selže. Klastr postavený podle tohoto principu může poskytovat vysokou odolnost proti chybám, ale v okamžiku vypnutí aktivního uzlu mohou být požadavky, které v tuto chvíli zpracovával, ztraceny [5,8].

- Teplé pohotovostní klastry

V teplém pohotovostním klastru aktivní uzly společně zpracovávají požadavky a v případě selhání jednoho nebo více uzlů se zátěž rozdělí mezi ostatní. Tento typ klastru lze také nazvat klastrem pro vyrovňování zatížení. Při použití tohoto klastru existuje také možnost ztráty dat zpracovaných uzlem, který selhal [5,9].

- Klastr s modulární redundancí

Pro tento princip platí, že všechny uzly v klastru zpracovávají stejné požadavky paralelně navzájem a po zpracovávání se vezme hodnota kteréhokoli z nich, který požadavek úspěšně dokončil. Takový systém zaručuje vyřízení požadavku bez ztráty dat. [5].

1.4 Klasifikace klastru

- Otevřený klastr

Každý uzel potřebuje mít přiřazený IP adresy a je nutné k nim mít povolený přístup přes internet nebo web. Tento typ klastru může způsobit zvýšené obavy ohledně zabezpečení [6].

- Zavřený klastr

Uzly jsou skryty za uzlem brány a poskytují zvýšenou ochranu. Potřebují méně IP adres a zvýší úroveň zabezpečení [6].

1.4.1 Komponenty klastrového počítače

- Uzly klastru
- Klastrový operační systém
- Hardware pro přepínání sítí
- Propojovací technologie a komunikační software

Klastry potřebovaly ke svému technologickému místu vývoj a technologii rychlého propojení, aby podporovaly mezi-procesorovou komunikaci s vysokou šířkou pásma a nízkou latencí mezi uzly klastru. Příklad pomalého propojení by představoval výkonnostní problémy a možnou chybovost při klastrových výpočtech. Dnes již vylepšená síťová technologie pomáhá realizovat výstavbu efektivnějších klastrů. Výběr technologie sítě pro propojení klastru závisí na několika faktorech, jako je kompatibilita s hardwarem a operačním systémem klastru, cenou a výkonem. Existují dvě metriky k měření výkonu pro propojení – šířka pásma a latence. Šířka pásma je množství dat, které může být přenášeno přes propojovací hardware v pevně stanoveném časovém období, zatímco latence je čas na přípravu a přenesení dat ze zdrojového uzlu do cílového uzlu [6].

1.5 Motivace klastrového systému

- Klastrový systém poskytuje relativně levná, nekonvenční řešení pro velké servery nebo sálové počítače
- Rychlejší způsobem řeší poptávku po kritičnosti obsahu a procesních službách
- Mnoho organizací a IT společností implementuje klastrové výpočty, aby rozšířily svou škálovatelnost, dostupnost, rychlost zpracování a řízení zdrojů za přijatelné ceny
- Zajišťuje, že výpočetní výkon je vždy k dispozici
- Poskytuje obecnou strategii pro implementaci a aplikaci paralelních vysoce výkonných systémů nezávislých na určitých dodavatelích hardwaru a jejich rozhodnutích o produktech [6]

1.6 Hrozby ohrožující provoz aplikace

Aplikace se stávají kritickými pro podnikání společností, což znamená, že nedostupnost služeb poskytovaných těmito aplikacemi vážně ovlivní finanční prosperitu společnosti. V

tomto ohledu se koncept přístupnosti stává základním aspektem provozu výpočetního centra. Příklady hrozeb pro provoz aplikace mohou být například [5].

- Zničení dat

Jeden z hlavních problémů v dostupnosti služby. Nejsnadnějším způsobem, jak se chránit, je pořizovat časté „snímky“ důležitých dat, aby bylo možné se kdykoli k poslední kopii vrátit.

- Selhání hardwaru

Výrobci hardwarových systémů (servery, disková úložiště) vyrábějí řešení s nadbytečnými komponentami – procesorovými deskami, řadiči systému, napájecími zdroji atd. Firmy tyto problémy řeší tím, že naddimenzují hardware na danou aplikaci. Například přiřadí čtyři jádra procesoru, kde stačí dvě nebo použijí nadbytek RAM paměti, kdyby jedna či více náhodou přestaly fungovat. V některých případech však selhání hardwaru může vést k nedostupnosti aplikace [5].

- Chyba aplikace

Chyba programátora v aplikaci, která již byla otestována na akceptaci a spuštěna do výroby, se může projevit v jednom případě ze stovek či tisíců případů. Pokud však k takové události přesto dojde, může to vést k okamžité ztrátě zisku v organizaci, například z důvodu zastavení zpracování transakcí a způsob odstranění chyby vyžaduje čas. Nejrychlejším řešením pravděpodobně bude nasadit předchozí verze aplikace [5].

- Konfigurační chyba

Správce provede změny v konfiguračních souborech, například DNS. Když otestuje DNS, služba zdá se být spuštěna a funkční, ale služba která DNS používá, například e-mail, začne mít problémy, které nejsou okamžitě zřejmé [5].

- Naplánovaná údržba

Výměna součástí, instalace aktualizací, restartování apod., představuje potencionální důvod nedostupnosti. Už při vstupu do počítačové RACK skříně musí být servisní technik opatrný. Každý zásah do serveru a odpojení jakékoli součásti se musí podařit bez vypojení či většího kontaktu s dalším kabelem, příp. jinou komponentou.

- Běžné problémy

I když organizace dělá vše pro ochranu před místními problémy, nemůže vždy zaručit dostupnost služby. Je proto nutné při plánování systému vzít v úvahu co nejvíce rizik [5].

1.7 Řešení hrozeb

K řešení závažných problémů mohou být k obnovení systému použity následující mechanismy [5]

- Záloha dat

Záloha dat je nejjednodušší, nejlevnější, ale také nejpomalejší při obnově systému. Data mohou být zálohována na pásková nebo disková média.

- Místní zrcadlení

Poskytuje dostupnost dat v reálném čase, a tudíž jsou data chráněna před zničením.

- Vzdálená replikace

Zde se předpokládá, že výpočetní místa jsou rozmístěna za účelem vytvoření kopie dat v rozmístěných datových centrech.

- Místní klastrování

Jakmile je ochrana dat zavedena, dalším krokem k zajištění dostupnosti aplikací je místní klastrování, tj. vytváření redundance hardwaru i softwaru.

- Vzdálené klastrování

Zde se opět předpokládá, že výpočetní místa jsou rozmístěna svou lokalitou za účelem provozuschopnosti například při výpadku elektřiny nebo lokální odstávce.

Myšlenka je jasná - čím více redundance zavedeme, tím vyšší jsou náklady na řešení, ale tím lépe jsou aplikace chráněny. Pro návrháře řešení je velmi důležité mít na paměti všechny tyto a novější technologie, protože pouze jejich správná kombinace a nastavení povede k vhodnému řešení úkolu stanoveného zákazníkem [5].

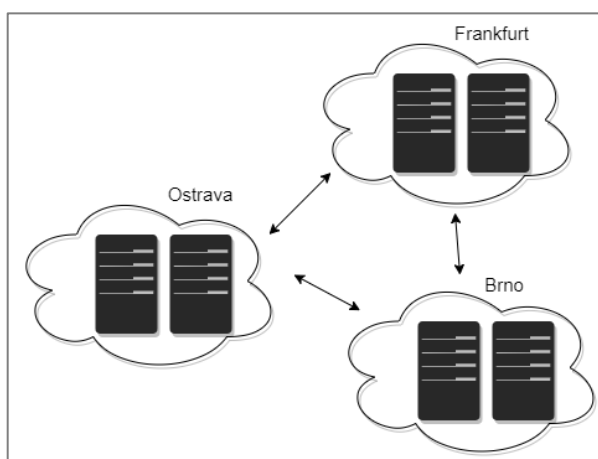
1.7.1 Geograficky distribuovaný klastr

Architektura klastru s vysokou dostupností, vytvořena a provozována v rámci jedné lokality, může chránit pouze před lokálními problémy. V případě technických či přírodních problémů postihujících celou lokalitu bude celý systém nepřístupný.

Dnes vzniká stále více úkolů, jejichž kritická povaha vyžaduje zajištění migrace služeb nejen v rámci lokality, ale také mezi geograficky rozptýlenými datovými centry. Při navrhování takových řešení je třeba vzít v úvahu nové faktory – vzdálenost mezi lokalitami, šířku pásma kanálů, jiného poskytovatele internetu, která replikace dat by měla být upřednostňována nebo třeba jaké použít protokoly. Úspěch projektu může záviset na řešení těchto problémů [5].

- Replikace dat z hlavní lokality na záložní

V případě selhání hardwaru nebo problému s aplikací či databází se software klastru nejprve pokusí přenést aplikační službu do jiného uzlu na primární lokalitě. Pokud se hlavní stránka z jakéhokoli důvodu stane nepřístupnou pro vnější svět, všechny služby, včetně DNS, migrují na záložní uzel, kde díky replikaci jsou již data přítomna. Služba je tedy pro uživatele obnovena [5].



Obrázek 1 Geograficky distribuovaný klastrový systém [zdroj: vlastní]

Nevýhodou tohoto přístupu jsou enormní náklady na nasazení dalšího teplého uzlu s kompletním vybavením a sítíovou infrastrukturou. Výhoda úplné ochrany však může tyto náklady převážet. Pokud centrální uzel není schopen dlouhodobě poskytovat služby, může vést k velkým ztrátám a dokonce ke konci podniku [5].

1.7.2 Výhody klastrového počítače

- Vysoký výkon

Systémy nabízejí lepší výkon než sálkové počítačové sítě.

- Snadná správa

Klastrový počítač je lehce ovladatelný, snadno implementovaný a je jednodušší ho odstavit než sálový počítač.

- Škálovatelnost

Do klastrových počítačů lze přidávat další zdroje. Může se jednat o hardwarovou škálovatelnost, kdy můžeme rozšířit výkon jednotlivých uzlů nebo rozšířit klastr a jeho funkčnost o nové zdroje, například přidáním další aplikace, kterou bude klastr řídit a provozovat.

- Rozšiřitelnost

Počítačové klastry lze snadno rozšířit přidáním dalších počítačů do sítě. Klastrový počítač je schopen kombinovat několik dalších zdrojů nebo sítí se stávajícím počítačovým systémem.

- Dostupnost

Vysoká dostupnost je souběžně s rostoucí závislostí na počítačích čím dál více vyhledávanější a v současné době má zásadní roli zejména ve společnostech, jejichž nejdůležitější funkcí je právě nabídka některé stabilní služby.

1.7.3 Nevýhody klastrového počítače

- Vysoká pořizovací cena

Nasazení klastru představuje velkou jednorázovou investici. Čím větší bude počet uzlů, tím větší bude potřebná infrastruktura a návrh takového systému bude mnohem komplikovanější a jeho zpracování dražší.

- Větší prostory

Infrastruktura se může zvýšit, protože ke správě, provozu a monitorování bude potřeba více serverů.

2 FAILOVER KLASTR

2.1 Sdílené úložiště

Failover klastry poskytují vysoce dostupné služby eliminací jednotlivých náznaků selhání a převzetím služeb z jednoho uzlu klastru do druhého v případě, že uzel přestane fungovat nebo odpovídat podle očekávání. Služby v klastru s vysokou dostupností obvykle čtou a zapisují data prostřednictvím připojených souborových systémů pro čtení a zápis. Klastr musí udržovat integritu dat, aby při převzetí kontroly nad službami jiným uzlem, pokračoval v práci a navázal na předchozí data jiného uzlu [10].

Red Hat poskytuje klastrování vysoké dostupnosti prostřednictvím své komponenty Pacemakeru, který bude popsán v kapitole [2.3].

2.1.1 Klastrové souborové systémy

Red Hat Global File System 2 (GFS2) je součástí doplňkového balíku Resilient Storage, který poskytuje klastrový souborový systém pro použití s doplňkem High Availability. GFS2 umožňuje více uzlům sdílet úložiště na úrovni bloku, jako by úložiště bylo připojeno lokálně ke každému uzlu klastru. Klastrový souborový systém GFS2 vyžaduje klastrovou infrastrukturu [11].

Logické svazky Logical Volume Management (LVM) jsou součástí doplňku High Availability a poskytují podporu pro LVM svazky ve dvou odlišných klastrových konfiguracích.

- Svazky LVM s vysokou dostupností (HA-LVM) v konfiguraci aktivního/pasivního převzetí služeb při selhání, ve kterých k úložišti přistupuje v jednu chvíli pouze jeden uzel klastru.
- Svazky LVM, které používají démona `lvmlockd` ke správě úložných zařízení v konfiguraci aktivní/aktivní. V této konfiguraci vyžaduje přístup k úložišti více než jeden uzel klastru současně. Démon `lvmlockd` je součástí doplňku Resilient Storage. Podpora `lvmlockd` také vyžaduje klastrovou infrastrukturu.

Sdílené logické svazky používající `lvmlockd` a HA-LVM jsou podobné v tom, že zabráňují poškození metadat LVM a jejich logických svazků, k čemuž by jinak mohlo dojít, pokud by více počítačů mělo přístup a mohlo přepisovat tyto data. Metadata jsou uložena jako ASCII a jedná se o podrobnosti konfigurace skupiny disků. HA-LVM se proti tomu chrání tak, že

logický svazek lze aktivovat pouze exkluzivně. To znamená, že je aktivní pouze na jednom počítači v daném čase. Zabránění režii koordinace klastru tímto způsobem zvyšuje výkon.

Sdílený svazek pomocí lvmlockd nemá tento přístup. Dovolí uživateli dle libosti aktivovat logický svazek na všech počítačích v klastru a zapisovat. Tato možnost si vynucuje použití ovladačů úložiště podporujících klastrové řešení (tzv. cluster-aware drivers), které umožňují souborovým systémům a aplikacím určit, který uzel má zrovna teď právo být „umístěný nad ostatními“, uzavřít si pro tento moment pro sebe svazek a zapisovat na něj. Svazek ale může být připojen k více počítačům [11,12].

Jak již bylo zmíněno, klastry jsou spravovány přes Pacemaker. HA-LVM i sdílené logické svazky jsou podporovány pouze ve spojení s klastry Pacemaker a musí být nakonfigurovány jako klastrové prostředky (resources).

2.2 Souborové systémy na Red Hat

Výběr souborového systému, který je vhodný pro danou aplikaci, je důležitým rozhodnutím kvůli velkému množství dostupných možností a souvisejících kompromisů. V této části budou popsány nejčastější na Red Hat.

2.2.1 Typy souborových systémů

Na nejobecnější úrovni lze souborové systémy dostupné na Red Hat seskupit do následujících hlavních kategorií

- Disk nebo místní souborový systém
- Síťový nebo souborový systém klient/server
- Sdílené úložiště nebo souborový systém sdíleného disku

2.2.2 Přehled lokálních souborových systémů

Místní (lokální) souborový systém se používá pro interní disky SATA nebo SAS, nebo když má server interní hardwarové řadiče RAID s místními disky. Tedy když souborový systém běží na jediném lokálním serveru a je přímo připojen k úložišti. Místní souborové systémy jsou také nejběžnějším souborovým systémem použitým na připojeném SAN (Storage Area Network) úložišti, když exportované zařízení SAN není sdíleno. SAN je dedikovaná datová síť, která slouží pro připojení externích zařízení k serverům (disková pole, páskové mechaniky a jiná zálohovací zařízení) [13].

2.2.2.1 Souborový systém XFS

XFS je robustní a vyspělý 64bitový žurnálový souborový systém, který podporuje velmi velké soubory a souborové systémy na jednom hostiteli. Je to výchozí souborový systém pro Red Hat Enterprise Linux 8. Žurnálování zajišťuje integritu souborového systému po zhroucení systému (například kvůli výpadku napájení), uchováváním záznamů o operacích souborového systému, které lze přehrát po restartování systému a souborový systém znovu namapovat [13].

XFS byl původně vyvinut na počátku 90. let společností SGI a má dlouhou historii provozu na extrémně velkých serverech a úložných polích. XFS podporuje velké množství funkcí, včetně následujících

- Zpožděná alokace
- Schopnost podporovat velké množství souběžných operací
- Rozsáhlá kontrola konzistence metadat za běhu
- Sofistikované algoritmy pro předcítání metadat
- Pevně integrované nástroje pro zálohování a obnovu
- Online defragmentace

Zatímco XFS je známý svou vysokou škálovatelností, v tabulce je uveden maximální podporovaný obraz souborového systému XFS společnosti Red Hat. Tyto limity jsou testovány týmem inženýrů a technické podpory Red Hat a je ověřeno plně funkční chování pro daný parametr [13,14].

Tabulka 1. Souborový systém XFS – certifikovaná velikost Red Hat [15]

Funkce	Red Hat 5	Red Hat 6	Red Hat 7	Red Hat 8
Maximální velikost souboru	100 TiB	100 TiB	500 TiB	8 EiB
Max. velikost souborového systému	100 TiB	300 TiB	500 TiB	1 PiB
ACL podpora	Ano	Ano	Ano	Ano

Access Control List (ACL) poskytuje mechanismus oprávnění pro souborové systémy. Je navržen tak, aby pomáhal s právy souborů v UNIX. ACL umožňuje udělit oprávnění jakémukoli uživateli nebo skupině k jakémukoli prostředku disku [16].

2.2.2.2 Souborový systém Ext4

Ext4 je čtvrtá generace souborového systému Ext. Ext4 jakožto 48bitový žurnálový souborový systém, který je výchozím souborovým systémem v Red Hat Enterprise Linux 6, může číst a zapisovat do souborových systémů Ext2 nebo Ext3. Formát souborového systému Ext4 není kompatibilní s ovladači Ext2 a Ext3 [17].

Ext4 však přineslo několik nových a vylepšených funkcí, které jsou běžné u většiny moderních souborových systémů a předchozím verzím chyběly.

- Snížení množství metadat díky spojitým rozsahům datových bloků
- Zpožděná alokace
- Kontrolní součet žurnálu
- Podpora velkého úložiště

Kompaktnějším a účinnějším způsobem sledování využitého prostoru v souborovém systému je použití metadat založených na rozsahu a funkce zpožděné alokace. Tyto funkce zlepšují výkon souborového systému a částečně snižují prostor, který metadata zabírají. Rozsahy snižují množství metadat potřebných ke sledování datových bloků u velkých souborů. Namísto ukládání seznamu každého jednotlivého bloku, který tvoří soubor, je myšlenkou uložit pouze adresu prvního a posledního bloku každého souvislého rozsahu bloků. Hlavní motivací pro rozsahy je spíše zlepšení výkonu (snížením fragmentace a menším počtem bloků metadat pro čtení a zápis) než úspora místa. Zpožděná alokace umožňuje souborovému systému odložit výběr trvalého umístění pro nově zapsaná uživatelská data, dokud nebudou data zapsána na disk. Větší a souvislejší alokace umožní souborovému systému potvrzovat rozhodnutí s mnohem lepšími informacemi a tím zvýšit výkon. [13,17,18].

Tabulka 2. Souborový systém Ext3 – certifikovaná velikost Red Hat [15]

Funkce	Red Hat 5	Red Hat 6	Red Hat 7	Red Hat 8
Maximální velikost souboru	2 TiB	2 TiB	2 TiB	2 TiB
Max. velikost souborového systému	16 TiB	16 TiB	16 TiB	16 TiB
ACL podpora	Ano	Ano	Ano	Ano

Tabulka 3. Souborový systém Ext4 – certifikovaná velikost Red Hat [15]

Funkce	Red Hat 5	Red Hat 6	Red Hat 7	Red Hat 8
Maximální velikost souboru	16 TiB	16 TiB	16 TiB	16 TiB
Max. velikost souborového systému	16 TiB	16 TiB	50 TiB	50 TiB
ACL podpora	Ano	Ano	Ano	Ano

- Výběr místního souborové systému

Pokud je server i úložné zařízení velké a není potřeba zmenšovat velikost souborového systému, XFS bude pravděpodobně nejlepší volbou. I s menšími úložnými poli funguje XFS velmi dobře, když je průměrná velikost souborů větší (například velikost stovek MB). Pokud stávající pracovní řešení fungovalo dobře s Ext3, migrace na Ext4 by měla aplikacím poskytnout vylepšené, ale přes to známé pracovní prostředí. Dvě klíčové výhody Ext4 oproti Ext3 na stejném úložišti zahrnují rychlejší kontrolu a opravy souborového systému a vyšší streamovací výkon při čtení a zápisu na vysokorychlostních zařízeních [13,14].

2.2.3 Sít'ové souborové systémy

Sít'ové souborové systémy, označované také jako souborové systémy klient/server, umožňují klientským počítačům přístup k souborům, které jsou uloženy na sdíleném serveru. To umožňuje více uživatelům na více počítačích sdílet soubory a prostředky úložiště. Takové souborové systémy jsou sestaveny z jednoho nebo více serverů, které exportují do jednoho nebo více klientů, sadu souborových systémů. Klientské uzly nemají přístup k základnímu

blokovému úložišti, ale spíše interagují s úložištěm pomocí protokolu, který umožňuje lepší řízení přístupu [13].

Nejběžnějším síťovým souborovým systémem klient/server pro Red Hat je Network File System (NFS). NFS umožňuje vzdáleným hostitelům připojit souborový systém přes síť a pracovat s těmito souborovými systémy, jako by byly připojeny lokálně. Red Hat poskytuje jak serverovou komponentu NFS, která se používá k exportu místního souborového systému přes síť, tak klienta NFS, který lze používat k importu těchto souborových systémů [13,19].

2.2.4 Souborové systémy sdíleného úložiště

Souborové systémy sdíleného úložiště, někdy označované jako souborové systémy klastru, poskytují každému serveru v klastru přímý přístup ke sdílenému blokovému zařízení přes místní SAN. Stejně jako výše uvedené souborové systémy klient/server fungují souborové systémy sdíleného úložiště na sadě serverů, které jsou všechny součástí klastru. Na rozdíl od NFS však žádný server neposkytuje přístup k datům nebo metadatům ostatním členům, ale každý člen klastru má přímý přístup ke stejnému sdílenému úložišti a všechny členské uzly klastru mají přístup ke stejné sadě souborů [13].

Koherence mezi-paměti je v klastrovém systému prvořadá kvůli zajištění konzistence a integrity dat. Musí existovat jednotná verze všech souborů v klastru viditelná pro všechny uzly v klastru. Aby se zabránilo členům klastru aktualizovat stejný blok úložiště ve stejnou dobu a způsobit poškození dat, používají souborové systémy sdíleného úložiště mechanismus širokého zamykání klastru k rozhodování o přístupu k úložišti, jako mechanismus kontroly souběžnosti. Například před vytvořením nového souboru nebo zápisem do souboru, který je otevřen na více serverech, musí komponenta systému souborů na serveru získat správný zámek [13].

Na sdíleném úložišti SAN se často používá protokol Small Computer Systems Interface (SCSI). Jedná se o soubor standardů pro fyzické připojení a přenos dat mezi počítači a periferními zařízeními. Standard SCSI definuje sady příkazů pro konkrétní typy periférií, přítomnost „neznámého“ jako jednoho z těchto typů ale značí, že jej lze použít teoreticky jako rozhraní pro jakékoli zařízení. V počátcích se SCSI používali pro pevné disky a páskové jednotky, ale může připojit širokou škálu dalších zařízení, včetně skenerů a jednotek CD.

Původní SCSI standard byl publikován už v roce 1986, od té doby ale přišla řada vylepšení a podpora stále rostoucí kapacity úložiště. Serial Attached SCSI (SAS) je sériové rozhraní

typu point-to-point, který do značné míry nahradil paralelní SCSI. Slouží k připojení pevných disků a páskových jednotek. Poprvé byl představen v 80. letech 20. století. Pro komunikaci používá standardní příkazy SCSI. SAS sběrnice nabízí výhody oproti starším paralelním technologiím. Kabele jsou tenčí a konektory méně objemné. Sériový přenos dat umožňuje použití delších kabelů než paralelní přenos dat. SAS nabízí zpětnou kompatibilitu se SATA 2.0 a novějšími. Díky tomu je možné připojit disky SATA na rozhraní SAS, avšak disky SAS k rozhraní SATA připojit nelze. [20,21].

V sítích SAN jsou disková pole spojena pomocí RAID, díky čemuž spousta pevných disků vypadá a funguje jako jedno velké úložné zařízení. Každé úložné zařízení nebo dokonce oddíl na tomto zařízení má přiřazené Logical Unit Number (LUN). Toto číslo je jedinečné v rámci SAN. Každý uzel v SAN, ať už je to server nebo jiné úložné zařízení, může přistupovat k uložišti odkazem na LUN. Konkrétnímu serveru nebo skupině serverů může být například udělen přístup pouze k určité části úložné vrstvy SAN ve formě LUN. Když úložné zařízení přijme požadavek na čtení nebo zápis dat, zkontroluje svůj přístupový seznam, aby zjistil, zda má uzel identifikovaný svou LUN a povolen přístup do úložné oblasti [10,22].

2.3 Pacemaker

Pacemaker je open source správce klastrových prostředků, tedy logika zodpovědná za životní cyklus nasazeného softwaru. Řídí prostředky a služby, které jsou spravovány vysoce dostupným klastrem.

Mezi klíčové vlastnosti Pacemakeru patří [23]

- Detekce a obnova poruch na úrovni uzlů a služeb
- Správa libovolných prostředků (vše, co lze skriptovat, může být prostředkem klastru)
- Podpora oplocení, také označováno pod zkratkou STONITH (popsáno později), pro zajištění integrity dat
- Podpora velkých i malých klastrů
- Podpora klastrů řízených kvorem
- Více možností konfigurace redundance
- Automaticky replikovaná konfigurace, kterou lze upravovat a aktualizovat z libovolného uzlu

2.3.1 Corosync

Corosync je open source program, který slouží k základním potřebám členství a komunikace mezi uzly pro klastry s vysokou dostupností. Corosync také spravuje pravidla kvora a poskytuje možnosti zasílání zpráv těm aplikacím, které pracují napříč více uzly v klastru, a proto musí mezi instancemi předávat stavové nebo jiné informace.

Corosync v podstatě umožňuje serverům komunikovat jako klastr, zatímco Pacemaker poskytuje možnost řídit, jak se klastr má chovat.

2.4 Architektura Pacemakeru

Na nejvyšší úrovni je Pacemaker klastr tvořen třemi částmi

- Komponentami, které nejsou přímo klastrované (tzv. non-cluster-aware), ale dávají Pacemakeru možnost a nástroj, jak řídit prostředky pomocí agentů. Tyto části jsou obsaženy v samotných prostředcích, jedná se o skripty, které umožní jejich spouštění, zastavení a sledování stavu, také místního démona, který maskuje rozdíly mezi různými standardy, které tyto skripty implementují. I když interakce těchto prostředků, v případě spuštění jako vícenásobné instance, může připomínat distribuovaný systém, stále jim schází mechanismy vysoké dostupnosti nebo autonomního klastrového řízení [23].
- Řízením prostředků pomocí skriptů v předchozím bodě. Pacemaker představuje mozek, který zpracovává a reaguje na události týkající se klastru. Mezi tyto události patří také připojení nebo odpojení uzlu z klastru, události služeb způsobené poruchami, údržbou nebo plánovanými činnostmi a dalšími administrativními úkony. Pacemaker vypočítá ideální stav klastru a začne mapovat cestu k jeho dosažení po jakékoli z těchto událostí. Tato cesta může zahrnovat přesun služeb, zastavování uzlů, a dokonce jejich vynucení do offline stavu pomocí vzdáleného vypnutí napájení [23].
- Poslední částí je již zmíněný projekt Corosync a podobné projekty jako CMAN nebo Heartbeat, které poskytují spolehlivé zasílání zpráv, informace o členství a kvoru v klastru [23]. Korum je vysvětleno v samostatné kapitole [2.4.4].

2.4.1 Součásti Pacemakeru

Klastr nakonfigurovaný pomocí Pacemakeru obsahuje velké množství démonů. Jsou to obslužné programy, které běží tiše na pozadí, monitorují a starají se o určité subsystémy a zajišťují, že klastr běží správně. Démony Pacemakeru monitorují členství v klastru, skripty,

které spravují služby a podsystémy správy prostředků, které monitorují různorodé prostředky [23].

- Klastrová informační základna (CIB)
- Démon pro správu sdílených zdrojů (CRMd)
- Démon pro správu místních zdrojů (LRMd)
- Modul zásad (PEngine nebo PE)
- Démon oplocení (STONITHd)

CIB představuje informačního démona Pacemakeru, který interně používá XML k distribuci a synchronizaci aktuálních konfiguračních a stavových informací z Designated Coordinatoru (DC) do všech ostatních uzlů klastru. DC je uzel přiřazený Pacemakerem, k ukládání a distribuci stavu a akcí prostřednictvím CIB. Obsah CIB se automaticky synchronizuje v celém klastru a používá ho PE k výpočtu ideálního stavu klastru a cest, jak ideálního stavu dosáhnout [23].

Akce s prostředky klastru jsou směřovány přes CRMd démona. Prostředky spravované CRMd mohou být dotazovány klientskými systémy, přesouvány, konkretizovány a v případě potřeby změněny [24].

Každý uzel klastru také obsahuje démona LRMd, který funguje jako rozhraní mezi CRMd a prostředky. LRMd předává příkazy z CRMd agentům prostředků, například spouštění, zastavování a předávání stavových informací [24].

DC provádí PE v požadovaném pořadí tak, že je předá buď démonu LRMd nebo CRMd peerům na jiných uzlech prostřednictvím infrastruktury zasílání zpráv klastru, která je zase předá již konkrétnímu LRMd. Všechny rovnocenné uzly hlásí výsledky svých operací zpět do DC a na základě očekávaných a skutečných výsledků buď provedou všechny akce, které je potřeba zpracovat a čekaly na dokončení předchozích, nebo přeruší zpracování a požádají PE o přepočítání ideálního stavu klastru na základě neočekávaných výsledků [24].

STONITH, zkratka pro „Shoot The Other Node In The Head“, je implementace oplocení Pacemakeru, chrání data před poškozením rozbitými uzly nebo souběžným přístupem. Funguje jako klastrový prostředek v Pacemakeru, který zpracovává požadavky oplocení, násilně vypíná uzly a odstraňuje je z klastru, aby byla zajištěna integrita dat. STONITH je konfigurován v CIB a lze jej monitorovat jako běžný prostředek klastru [24].

STONITH hraje roli v případě, kdy při přepínání na jiný uzel, klastrovanou službu na aktivním uzlu nelze zastavit. V tomto případě Pacemaker využije implementace STONITH a přinutí celému uzlu jít do offline stavu a násilně tak přestat poskytovat služby. Následně Corosync označí uzel za offline a Pacemaker spustí služby na jiném uzlu [24].

2.4.2 Klastrové prostředky

Klastrové aplikace jsou navrženy speciálně pro použití v klastrovém prostředí. Vědí o existenci dalších uzlů a jsou schopny s nimi komunikovat. Klastrová databáze je jedním příkladem takové aplikace. Instance klastrové databáze běží v různých uzlech a musí upozornit ostatní uzly, pokud potřebuje uzamknout nebo upravit některá data. Webový server je také typickým příkladem aplikace klastru. V tomto případě všechny uzly v klastru mají stejný obsah a klientovi je jedno, ze kterého serveru se požadovaný obsah poskytuje. Klastrové prostředky jsou abstrakcí nahrazeny agenty. Klastr nemusí rozumět tomu, jak prostředek funguje, protože spoléhá na agenta prostředku, že udělá správnou věc, když dostane příkaz ke spuštění, zastavení nebo sledování služeb. Z tohoto důvodu je klíčové, aby agenti prostředků byli dobře otestováni. Konfigurace agentů prostředků bývá obvykle ve formě skriptů shellu. Mohou však být napsány pomocí jakékoli technologie (C, Python, Perl), která autorovi vyhovuje [24].

2.4.3 Třídy prostředků

Pacemaker podporuje několik tříd agentů

- Open Cluster Framework (OCF)

Standard OCF je v podstatě rozšířením konvencí LSB (Linux Standard Base) pro init skripty na podporu parametrů. Umožňuje aplikacím, aby byly jejich stavy popsitelné a rozšiřitelné. Specifikace OCF mají přesné definice výstupních kódů, které musí akce vracet. Klastr dodržuje tyto specifikace a zadání nesprávného výstupního kódu způsobí, že se klastr bude minimálně chovat jiným způsobem. Klastr zejména potřebuje rozlišit zcela zastavený prostředek, od prostředku, který je v nějakém chybném a neurčitěm stavu.

Parametry jsou předávány agentovi prostředků jako proměnné se speciálním prefixem OCF_RESKEY. Takže parametr, který uživatel považuje za ip, bude předán agentovi prostředků jako OCF_RESKEY_ip.

Třída OCF je nejpreferovanější, protože jde o průmyslový standard, vysoce flexibilní a samo popisující [24].

- LSB

Agenti prostředků LSB abstrahují systémové služby LSB, které se nacházejí v /etc/init.d. Agenti LSB obvykle mají jen návratové kódy pro úspěšné a neúspěšné stavy.

- Systemd

Některé novější distribuce nahradily starý styl inicializačních démonů a skriptů „SysV“ alternativou nazvanou Systemd. Pacemaker je schopen tyto služby spravovat, pokud jsou k dispozici. Místo init skriptů má systemd soubory jednotek. Obecně platí, že soubory jednotek poskytuje distribuce OS, ale existují online průvodci pro převod z init skriptů [25].

- Systémové služby

Jelikož existují různé typy systémových služeb (Systemd, LSB...), Pacemaker podporuje speciální alias služby na rozpoznání, jaká ze systémových služeb se vztahuje na daný uzel klastru. To je zvláště užitečné, když klastr obsahuje kombinaci všech systémových služeb.

- Oplocení

Oplocení a třída STONITH byla diskutována výše.

- Nagios pluginy

Nagios pluginy nám umožňují sledovat služby na vzdálených hostitelích. Pacemaker je schopen provádět vzdálené monitorování pomocí zásuvných modulů, pokud jsou k dispozici.

Běžným případem použití je nakonfigurovat nagios pluginy jako prostředky patřící do kontejneru prostředků (obvykle virtuálního počítače) a kontejner bude restartován, pokud některý z nich selže. Dalším využitím je konfigurovat je jako běžné prostředky, které mají být použity pro monitorování hostitelů nebo služeb prostřednictvím sítě [23].

2.4.4 Korum

Aby byla zachována integrita a dostupnost klastru, používají klastrové systémy koncept známý jako korum. Klastr má korum, když je více než polovina uzlů klastru online. Ke zmírnění možnosti poškození dat v důsledku selhání, Pacemaker ve výchozím nastavení zastaví všechny prostředky, pokud klastr nemá korum.

Kvorum je ustaveno pomocí hlasovacího systému. Když uzel klastru nefunguje tak, jak by měl, nebo ztratí komunikaci se zbytkem klastru, většina pracovních uzlů může hlasovat pro izolaci a v případě potřeby sestřelit (STONITH) uzel kvůli plynulosti provozu. Například v klastru se šesti uzly je kvorum ustaveno tak, že musí fungovat alespoň čtyři uzly klastru. Pokud většina uzlů přejde do režimu offline nebo se stane nedostupným, klastr již nemá kvorum a Pacemaker zastaví klastrové služby a akce. Funkce kvora v Pacemakeru zabraňuje tomu, co je také známé jako split-brain, jev, kdy je klastr oddělen od komunikace, ale každá část nadále funguje jako samostatný klastr. Podle provedených akcí zapisují data a můžou způsobit poškození nebo ztrátu dat [24].

2.4.5 Oplocení [fencing]

Pokud selže komunikace s jedním uzlem v klastru, pak ostatní uzly v klastru musí mít možnost omezit nebo uvolnit přístup k prostředkům, ke kterým může mít přístup nefunkční uzel klastru. Toho nelze dosáhnout kontaktováním samotného uzlu klastru, protože uzel klastru nemusí reagovat. Místo toho se musí využít externí metoda, která se nazývá oplocení pomocí plotového agenta. Zařízení plotu je externí zařízení, které může klastr použít k omezení přístupu ke sdíleným prostředkům chybným uzlem nebo k provedení tvrdého restartu, či vypnutí uzlu klastru [24].

Bez nakonfigurovaného oplocení není spolehlivý způsob, jak zjistit, že prostředky dříve používané odpojeným uzlem klastru byly uvolněny, a to by mohlo bránit spuštění služeb na kterémkoli z ostatních uzlů klastru. Naopak systém může chybně předpokládat, že uzel klastru uvolnil své prostředky, což může vést k poškození a ztrátě dat. Bez nakonfigurovaného zařízení oplocení nelze zaručit integritu dat a konfigurace klastru nebude podporována.

Když probíhá oplocení, není povoleno spuštění žádné jiné operace klastru. Normální provoz klastru nelze obnovit, dokud není dokončeno oplocení nebo dokud se uzel klastru znovu nepřipojí ke klastru po restartování uzlu [24].

II. PRAKTICKÁ ČÁST

3 LINUX

Prvním krokem k nakonfigurování failover klastru na Linuxu je výběr Linuxové distribuce. Linuxová distribuce, často uváděna zkráceně jako Linux distro, je operační systém sestavený z komponent vyvinutých různými open source projekty a programátory. Samotný pojem Linux však označuje pouze jádro, přestože se tento zkrácený název používá pro Linux distribuce. Na jedné straně každá nová distribuce vychází a těží z práce odvedené ostatními, zatímco na druhé straně často dochází k nevraživosti a konfliktům mezi distribucemi a jejich komunitami. Každá distribuce obsahuje linuxové jádro (základ operačního systému), obslužné programy GNU shell (terminálové rozhraní a příkazy), X server (pro grafické rozhraní počítače), desktopové prostředí, systém správy balíčků, instalační program a další služby. Vzhledem k tomu, že se jedná o software s otevřeným zdrojovým kódem, může si kdokoli vytvořit svou vlastní linuxovou distribuci tím, že si ji sestaví ze zdrojového kódu sám, nebo úpravou existující distribuce [26].

Existují komerčně podporované distribuce, jako je Fedora (Red Hat), openSUSE (SUSE) a Ubuntu (Canonical Ltd.), a zcela komunitně řízené distribuce, jako je Debian, Slackware, Gentoo a Arch Linux. Většina distribucí je tzv. „ready to use“ (připravena k použití) a přednastavena pro konkrétní instrukční sadu, zatímco některé distribuce jsou distribuovány ve formě zdrojového kódu a kompilovány lokálně během instalace. Některé komerční distribuce účtují uživatelům poplatek za podporu a za další služby, které nabízí zákazníkům. Licence s otevřeným zdrojovým kódem však zakazuje zpoplatnění softwaru s otevřeným zdrojovým kódem. Linux naplnil velké očekávání, které mělo spoustu lidí, kteří se podíleli na tomto projektu. Linux dnes má spoustu distribucí, které se zrodily díky všestrannosti, síle, komunitě a lidem, kteří nadále udržují toto jádro [26].

Linuxových distribucí existuje velké množství a stále jich přibývá. Existují a tvoří se další, protože ani jedna distribuce nemůže uspokojit přání každého uživatele na planetě. Odlišní lidé mají rádi různé způsoby a mají různé návyky, jak věci dělat. Nejen to, distribuce, která je navržena pro použití na serverech, nemusí nutně vyhovovat notebooku. Takže naštěstí existují tisíce distribucí, ze kterých je možno vybrat.

První oficiální distribuce vznikla v únoru 1992 a jmenovala se MCC Interim. Byla to první distribuce, kterou bylo možné nainstalovat na počítač, dodávající linuxové jádro s uživatelským obslužným systémem GNU. Ve stejném roce byla vytvořena nová (a v té době populární) distribuce nazvaná Softlanding Linux System (známá jednoduše jako SLS), díky

které zase vznikl dnes známý Slackware, vytvořený Patrickem Volkerdingem. Slackware dodnes zůstává nejstarší dochovanou distribucí Linuxu [27].

V době, kdy Slackware přišel na scénu, existovalo již pár linuxových distribucí. O několik měsíců později, 16. srpna 1993, se však objevil Debian. Jedna z nejdůležitějších distribucí, která dnes nese pomyslnou korunu za nejstarší dochovanou nezávisle vyvinutou linuxovou distribuci. Debian nebyl klonem (forkem) žádné předchozí práce, ale samostatným projektem, který vytvořil Ian Murdock. Debian je zcela řízen komunitou a zůstává největším nekomerčním distributorem Linuxu [27].

Téměř rok po zrodu Debianu přišel v roce 1994 na scénu třetí a poslední člen nejvlivnějších distribucí, Red Hat Linux. Tato distribuce byla původně vytvořena Marcem Ewingem, ale krátce poté se spojila se společností Boba Younga, ACC Corporation, a vytvořila Red Hat Software. Od samého začátku byl Red Hat Linux navržen s ohledem na podnikový svět. Red Hat distribuce byly a jsou komerční implementací distribuce Linuxu, postavené na svobodném softwaru [27].

3.1 Slackware

Patrick Volkerding vytvořil distribuci Slackware a ovládá ji do dnes. Má kolem sebe tým a za ním oddanou komunitu, ale je to on, kdo řídí, jaké kroky bude tato distribuce dělat.

Samotná distribuce se vyznačuje jednoduchostí a snaží se zůstat co nejvíce unixová. Své balíky příliš neopravuje, spíše zasílá produkty, které se co nejvíce podobají upstreamu. Slackware ponechává na uživateli kontrolu a příliš nevyužívá správce balíčků a i když může instalovat, upgradovat a odstraňovat balíčky, nesleduje ani nespravuje závislosti. Tento úkol spadá do pravomoci správci systému (nebo uživatelům) a je jedním z nejvýraznějších rozdílů mezi Slackwarem a následujícími dvěma. Z těchto důvodů může být považován za hůře použitelnou nebo také za distribuci pro pokročilejší, ale fanoušci distribuce v tom vidí nezbytné nástroje síly a flexibility. Slackware je mezi komunitou velmi uznávaný a stabilní [28].

Oficiálně podporuje pouze jedno hlavní desktopové prostředí, KDE, ale komunitou jsou podporovány i další, například GNOME. Zatímco mnoho dalších novějších distribucí přidává další vrstvy komplexnosti, aby věci usnadnilo, Slackware zůstává věrný svým UNIX kořenům a nabízí jednoduchý, ale výkonný a vysoce konfigurovatelný systém [28].

3.2 Debian

Debian se skládá z celosvětové komunity dobrovolníků, včetně více než tisíce vývojářů, kteří společně pracují na vytvoření nejlepšího možného operačního systému ze svobodného softwaru. Debian je jedinečný v tom, že se řídí svou ústavou, společenskou smlouvou, směrnicemi pro svobodný software a politickými dokumenty. Struktura organizace jako taková je velmi oficiální, se zvoleným vedoucím, tajemníkem a technickým týmem. Volby vedení se konají každý rok [28].

Na rozdíl od Slackware nemá vedoucí Debianu absolutní moc. Ve skutečnosti prostřednictvím obecného usnesení mohou vývojáři zvrátit rozhodnutí, odstranit vůdce a dokonce provést změny v ústavě. Vývojáři mohou také hlasovat o důležitých otázkách ovlivňujících projekt [28].

Debian se pyšní plně rozvinutým systémem správy balíčků, který obsahuje několik důležitých komponent. Tento systém nejenže provádí očekávané úkoly, jako je instalace a odstraňování balíčků, ale také automaticky zpracovává závislosti. Toto byla přelomová komponenta Debianu, která přišla velmi brzy, čímž se odlišovala od všech ostatních distribucí své doby. Debian používá formát balíčků .deb [28].

Všechna vydání Debianu jsou pojmenována po postavách z filmu společnosti Pixar, Toy Story. Debian je známý vysokou kvalitou svých vydání. Projekt udržuje tři hlavní větve, stabilní, testovací a nestabilní (tzv. Sid – postava z filmu). Ačkoli oficiální desktop je GNOME, projekt podporuje téměř všechny existující správce desktopů a oken. To je další ostrý kontrast ke Slackware, který oficiálně podporuje pouze KDE [28].

3.3 Red Hat

Red Hat Linux byl od začátku o komercializaci Linuxu. Za tímto účelem zaznamenal velký úspěch prodejem služeb ve formě předplatných pro podporu, školení a integraci. Popularitě vděčí za to, že je široce používán jako podporovaná distribuce Linuxu ve firemním prostředí. K dispozici jsou stovky kurzů Red Hat Linux a na dlouhou dobu byl „Linux“ často synonymem pro „Red Hat“ [28].

Jeho distribuce Red Hat Enterprise Linux (RHEL) je k dispozici pouze v binární formě při zakoupení od Red Hat, na rozdíl od našich dvou dalších distribucí. Celý zdrojový kód operačního systému je však přístupný zcela zdarma a díky tomu vzniká mnoho dalších distribucí (např. CentOS) [28].

Většinu vývojových prací na RHEL provádějí zaměstnanci Red Hat. Na rozdíl od Slackware a Debianu, je vývoj této distribuce primárně založen na změnách v komunitní distribuci Fedore sponzorované Red Hatem. Ačkoli jde o stabilní a perspektivní distribuci samo o sobě, Fedora je v podstatě testovací základnou pro novou technologii, která si najde cestu do komerčních nabídek Red Hat.

Na rozdíl od Debianu získal Red Hat komplexní systém správy balíčků poměrně pozdě. Ve svém jádru využívá balíčků ve formátu RPM, které jsou zpracovávány různými nástroji nízké úrovně. Dnes je instalace balíčků na Red Hat na stejné úrovni jako na Debianu, s podporou sledování závislostí a mnoha efektními funkcemi pro zajištění konzistentního stavu [28].

Red Hat vydal patentovou politiku, kterou využívá ve prospěch svobodného softwaru. Jejich obchodní model zaznamenal obrovský úspěch a je často používán jako ukázkový příklad schopnosti vydělávat peníze ze svobodného softwaru [28].

Red Hat dlouhodobě poskytuje jeden z nejspolehlivějších operačních systémů pro Linuxové firemní prostředí. Mezi jeho hlavní vlastnosti patří: [29]

- Nativní nástroje pro nasazení, vývoj a automatizaci kritických pracovních zátěží
- Zabezpečení včetně celo-systémových kryptografických zásad, auditování a příznaků kompilátoru
- Široká integrace s podnikovými aplikacemi, jako je PostgreSQL, SQL Server a SAP HANA
- Infrastruktura kontejnerů a nástroje pro vývoj a nasazení kontejner balíčků
- Podpora pro pracovní zátěže napříč hybridním cloudem, virtualizovanými prostředím a prostředím na fyzické bázi

3.3.1 RHEL

RHEL je open source distribuce Linuxu vyvinutá společností Red Hat pro komerční využití. Je založen na Fedore, což je komunitní projekt. Obrovské množství softwaru, které je k dispozici na RHEL, je nejprve vyvinuto a otestováno na Fedore, než bylo vydáno jako oficiální, což dokazuje jeho integritu a spolehlivost.

3.3.1.1 RHEL 8

Red Hat Enterprise Linux 8, jak název napovídá, je osmá major verze distribuce RHEL vyvinutá společností Red Hat a dodávaná zákazníkům prostřednictvím licenčního modelu.

Díky spolehlivosti, přísnosti testů a zajištění stability, kterou má na starost společnost Red Hat je RHEL 8 skvělým nástrojem pro produkční úlohy. Velká část společností po celém světě používá tuto distribuci a ukázalo se, že funguje velice dobře. V případě jakýchkoli problémů poskytuje Red Hat placenou podporu, kterou mohou zákazníci využít k vyřešení problémů a udržení obchodní kontinuity. Navíc nabízejí certifikace, díky nimž jsou jejich kvalifikovaní správci důvěryhodní po celém světě [30].

Výhody

- Podpora Red Hat, Inc.
- Nejnovější stabilní verze softwaru
- Stabilní a vynikající zejména pro webové aplikace
- Krokové vydání aktualizací zajišťuje, že vše, co vydají, prošlo přísným testováním
- Je důvěryhodný a výkonný
- Stabilní operační systém podle světových standardů
- Zdrojový kód je poskytnut
- Certifikace pro správce, aby bylo zajištěno, že se systémy zachází vyškolený personál

Nevýhody

- Jedná se o podnikovou verzi, a proto je nutné licencování
- Licenční model se liší od většiny distribucí
- Školení a dokumentace jsou poměrně drahé
- Podpora je placená
- Upřednostňuje stabilitu před novými technologiemi

3.4 Red Hat distribuce

Nyní dochází ke změně v Red Hat distribucích, kdy z podnikového, stabilního operačního systému CentOS Linux se stává vývojová distribuce pro RHEL, CentOS Stream. CentOS byl dlouhou dobu nejlepším RHEL forkem a přinášel Linux na produkční úrovni ve formě předchozí verze RHEL, která již byla otestována, zbavena chyb a patřičně zdokumentována.

Tabulka 4 Vývojová linie RHEL distribucí [zdroj: vlastní]

Do roku 2021	Od roku 2022
Fedora	Fedora
Red Hat	CentOS Stream
CentOS	Red Hat

CentOS tímto zanechal mezeru na trhu a dal příležitost novým projektům, které budou mít stejný směr a cíl, který měl donedávna ještě CentOS.

3.4.1 CentOS Stream 8

Koncem roku 2020 projekt CentOS oznámil, že mění způsob, jakým bude pokračovat. Byla to šokující zpráva, protože přesunuli CentOS Linux, distribuci pyšící se stabilitou, jakou má samotný RHEL, na postupně vydávanou distribuci Linux CentOS Stream. To znamená, že CentOS Stream bude vycházet vždy těsně před nějakou aktuální verzí RHEL. Umístěn bude jako střední proud mezi Fedorou a RHEL. Pro zjednodušení to znamená, že CentOS již nebude stabilní distribuce, ale průběžně vydávaná linuxová distribuce nyní známá jako CentOS Stream 8. Pokud jde o běžící CentOS 8, Red Hat převzal zaštitění aktualizací do konce roku 2021, což přivedlo mnoho uživatelů CentOS ke zděšení, protože se již dříve přizpůsobili a naklonili příslibu, že CentOS bude dostávat aktualizace až do roku 2029 [30,31].

Průběžně vydávaná distribuce je ta, která je neustále aktualizována, a proto může dojít k vážnému problému s provozem nebo zabezpečením produkčních úloh. Je to proto, že se něco může v průběžném vydání změnit na úkor aplikací. Použití jej v produkci lze jen s vědomím, že se chyby mohou kdykoli objevit, což je v produkci nežádoucí [30].

Výhody

- Podpora Red Hat
- Nabízí nejnovější verze softwaru

Nevýhody

- Nejasná stabilita a spolehlivost
- S průběžnou verzí se mohou v produkčním systému objevit i větší chyby

3.4.2 AlmaLinux

AlmaLinux OS je open source, který zaplňuje mezeru po ukončení stabilní verze CentOS Linux. AlmaLinux OS je 1:1 binárně kompatibilní fork RHEL řízený a vytvořený komunitou.

Jako samostatný, zcela bezplatný operační systém se AlmaLinux OS těší ročnímu sponzorství ve výši 1 milionu USD od společnosti CloudLinux Inc a podpoře od dalších sponzorů. Pokračující úsilí o rozvoj řídí členové komunity. AlmaLinux OS je serverový operační systém podnikové třídy a stabilní distribuce Linuxu s pravidelnými verzemi, které přicházejí s dlouhými okny podpory [30,32].

Výhody

- Je plně podporován a slibuje se, že bude vždy zdarma
- Distribuce je podporována investicemi a dlouhodobými závazky podpory, aby byla zajištěna distribuce bez omezení a poplatků
- Poskytuje nástroje a dobrou dokumentaci k tomu, jak dosáhnout přechodu z CentOS na AlmaLinux
- Komunita se stále rozrůstá
- Existuje komerční podpora pro ty, kteří by se do vývoje chtěli pustit taky

Nevýhody

- Ve srovnání s jinými stabilními distribucemi vznikla nedávno

3.4.3 Rocky Linux 8

Rocky Linux je další open source distribuce, která je pod intenzivním vývojem komunity. Rocky Linux je veden Gregory Kurtzerem, zakladatelem projektu CentOS. Rocky Linux 8 je následnou přestavbou RHEL 8 a jako takový je v souladu s životním cyklem RHEL 8, a proto bude aktivně udržován až do roku 2029. Rocky Linux si zatím získal nesmírné přijetí a podporu komunity a jeho budoucnost vzkvétá. Je založen na stabilním zdrojovém kódu RHEL, a proto jej lze použít jako OS k produkčním úlohám bez jakýchkoli problémů a pochybností [30,32].

Výhody

- Dobrá náhrada za CentOS 8
- Podporováno velice živou a rostoucí komunitou, která touží napravit škody, kterou způsobila změna v logice CentOS
- Nabízí nejnovější verzi softwaru

Nevýhody

- Ve srovnání s jinými stabilními distribucemi také vznikla tato distribuce nedávno

3.4.4 Souhrn z distribucí

Porovnání distribucí není nikdy snadný úkol, protože existuje mnoho funkcí a vlastností, které mohou toto porovnání mnohem více natáhnout a zkomplikovat výběr. V této práci je uvedeno pouze několik základních vlastností a od úvodu této kapitoly je zaměření směřováno jen na distribuce podobné RHEL. Praktická část byla zhotovena na distribuci AlmaLinux OS. Tento nový projekt binárně kompatibilní s RHEL totiž slibuje dlouhou podporu a hlavně bez nutnosti za ní platit.

3.5 Virtualizace

Virtualizace vytváří softwarovou iluzi fyzických prostředků, ať už na úrovni jednotlivých komponent nebo celého počítače. Což umožňuje vytvářet více nezávislých prostředí při sdílení jednoho fyzického zdroje.

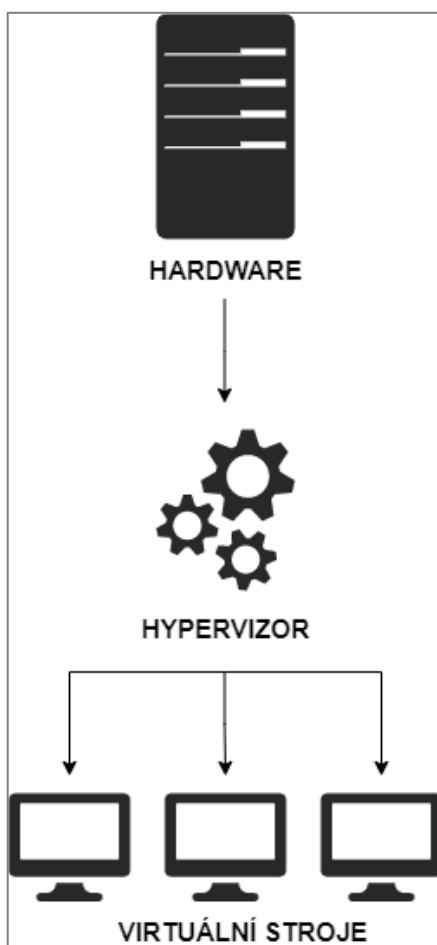
3.5.1 Typy virtualizace

Základní software, jenž umožňuje provoz VM je označován jako hypervizor nebo také Virtual Machine Monitor. Vytváří vrstvu mezi virtuálními stroji a hardwarovou vrstvou

fyzického počítače, zároveň udržuje oddělené výpočetní prostředí pro jednotlivé hostující systémy. V závislosti na tom, zda běží přímo na fyzické vrstvě nebo až v prostředí operačního systému jsou rozlišovány dva typy tzv. nativní nebo hostovaný [33].

3.5.2 Typ 1 (nativní, plný)

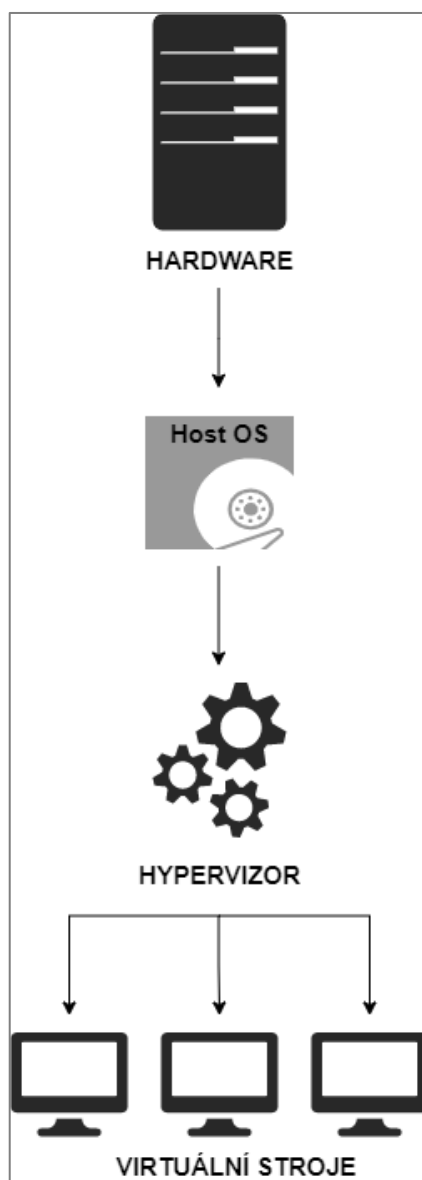
V anglické literatuře bývá označován také jako bare metal (holé železo), což napovídá, že je spuštěn bezprostředně na hardwaru, nahrazuje operační systém a má přístup přímo k fyzickým prostředkům. Ty následně poskytuje hostujícím virtuálním strojům. Kód je vykonáván v procesoru hostujícího systému napřímo. Toto s sebou nese požadavek kompatibility instrukční sady hostovaného a hostujícího systému. Hypervizor typu 1 vyžaduje podporu hardwarové akcelerace, která umožňuje provádět složité operace spojené se správou virtuálních prostředků. Tento typ nabízí nejlepší využití výkonu hardwaru, a proto nachází uplatnění především v serverovém prostředí. Mezi významné technologie využívající tento přístup patří Microsoft-Hyper-V, KVM nebo VMware vSphere. Tento typ by byl pravděpodobně v praxi použitý i pro tento projekt [33].



Obrázek 2 Virtualizace - typ 1 [zdroj: vlastní]

3.5.3 Typ 2 (hostovaný)

Je spouštěn v prostředí operačního systému, jako softwarová vrstva nebo aplikace, tudíž nemá přímý přístup k hardwaru. Všechny aktivity musí být zpracovány přes hostitelský operační systém a dochází k určité ztrátě výkonu. Takové řešení ale může být výhodné v případě, kdy uživatel chce provozovat více operačních systémů na jednom osobním počítači. Typickými nástroji s hypervizorem toho typu jsou VMware Workstation nebo Oracle VirtualBox [33]. Tato práce byla zhotovena na Oracle VirtualBoxu, protože je to multiplatformní virtualizační nástroj distribuovaný jak pro Linux/Unix, tak pro Windows a MAC OS. VirtualBox ve verzi 6.1.32 je snadno ovladatelný, spolehlivý a na účely tohoto školního projektu dostačující.

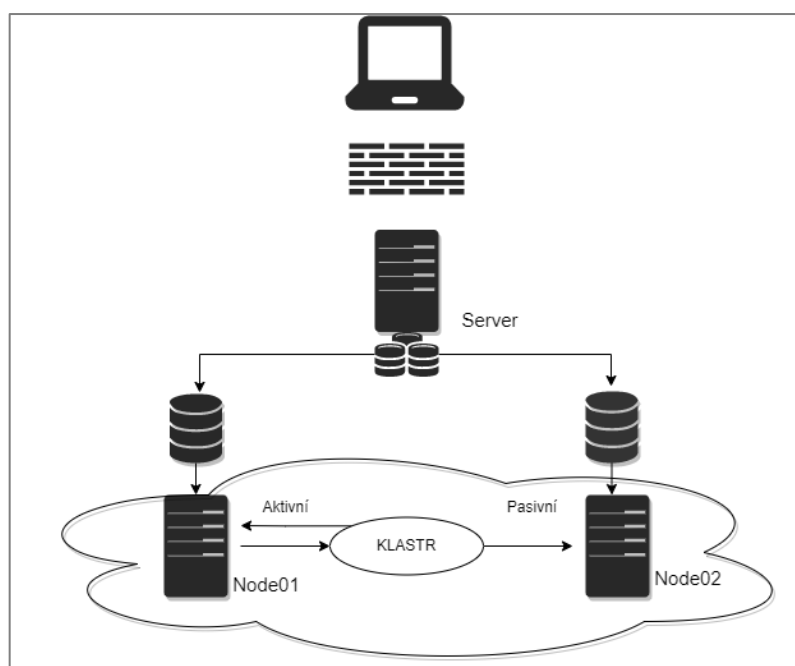


Obrázek 3 Virtualizace - typ 2 [zdroj: vlastní]

4 STUDENÝ POHOTOVOSTNÍ KLASTR

Praktická část byla zahájena spuštěním tří virtuálních strojů. Před spuštěním nového virtuálního stroje, je potřeba ověřit kolik hardwarových prostředků fyzický stroj obsahuje a kolik z nich je možno virtuálním strojům přiřadit. Serverová edice Red Hat podporuje virtualizaci a lze ji provozovat na virtuálním počítači buď v hostiteli, jako virtualizovanou, na emulátoru nebo v hypervizoru, jako je VirtualBox. Serverová edice Red Hat (v tomto případě její stream AlmaLinux OS) byla nainstalována formou minimální instalace a tedy nasazení jen malé sady potřebného softwaru k sestavení serveru. Tato edice navíc používá režim zobrazování bez grafické podpory, což umožní využívat a alokovat méně systémové paměti. Minimální požadavky na použitelný systém, jsou 512 MB systémové paměti, 5 GB místa na disku a je dobré mít také 2 síťové adaptéry (jeden NAT a druhý hostitelský).

Každý virtuální stroj byl vybaven 1 GB operační pamětí, 16 MB na video paměť a 1 virtuálním jádrem procesoru. Úložiště virtuálních strojů bylo navrženo tak, že každý dostal přiřazen svůj pevný disk s rozhraním SATA o velikosti 20 GB, na kterém byl umístěn systém a sloužil jako primární úložiště. Na Serveru, jak bude označován první instalovaný a konfigurovaný virtuální stroj, byl navíc připojen 8 GB disk, který bude sloužit jako sdílené úložiště pro aplikační uzly Node01, Node02. Sdílené úložiště SAN bude připojeno na aplikační uzly pomocí protokolu iSCSI, což je modernější verze již zmiňovaného protokolu SCSI.



Obrázek 4 Schéma virtuálních strojů [zdroj: vlastní]

Po dokončení bude HA klastr složen ze dvou uzlů, používajících operační systém AlmaLinux OS 8.5, nakonfigurován v aktivní/pasivní konfiguraci. Toho bude dosaženo nasměrováním virtuální IP [VIP] adresy, která odkazuje na aktivní uzel. Toto řešení poskytuje bezpečný způsob, jakým bude umožněno přistupovat k webové službě. Uzel, který po nastartování klastru přebere VIP adresu, ji bude hostit, až dokud nebude zjištěno selhání. V případě, že Pacemaker zjistí, že aktivní uzel je nedostupný, spustí na pasivním uzlu skripty a začne přesouvat potřebné prostředky tak, aby si pasivní uzel přebíral klastrovanou VIP adresu a služby. Následný síťový provoz, který bude směřován na VIP adresu, bude tedy směřován na pasivní uzel, který převeze roli a funkci aktivního uzlu a bude zpracovávat příchozí provoz.

4.1 Instalace a konfigurace Serveru

Po instalaci systému se doporučuje provést aktualizace na všech strojích, aby bylo ověřeno, že systém používá aktuální balíky. Pravděpodobně se nainstaluje i nový kernel, a proto bude vhodné stroje restartovat.

Po instalaci systému je v `/etc/yum.repos.d` pět přichystaných souborů s více jak dvaceti repositáři. Z těchto repositářů jsou aktivní pouze tři, a to BaseOS, AppStream a Extras v konfiguračním souboru `almalinux.repo`.

Síť byla nakonfigurována pomocí služby `network-scripts`. Volba konfigurace síťového rozhraní se může u různých správců systému lišit. Od RHEL 8 se preferuje použití `NetworkManager` a jeho utilit `nmcli` a `nmtui` k zobrazení a správě síťového rozhraní. Službu `network-scripts` lze doinstalovat pomocí `yum install network-scripts`. Konfigurace se provádí ruční úpravou souborů `ifcfg` (interface config). Podle konvence se pojmenovává konfigurační soubor `ifcfg-zařízení` a řetězec zařízení obsahuje stejnou direktivu jako `DEVICE` v samotném konfiguračním souboru. Konfigurace pro rozhraní `enp0s3` bude tedy soubor v `/etc/sysconfig/network-scripts`, který se nazývá `ifcfg-enp0s3` a bude obsahovat mimo jiné i parametr `DEVICE=enp0s3`.

4.1.1 Sdílené úložiště

Pro vytvoření sdíleného úložiště bylo použito iSCSI. Na serveru, který bude fungovat jako iSCSI target, je připojen disk `sdb`, který bude exportován virtuálním uzlům `Node01` a `Node02`, neboli iSCSI initiators. Export se realizuje přes utilitu `targetcli`, rozhraní je

intuitivní a je možné využívat tabulátoru k doplnění příkazů. Targetcli se spouští stejnojmenným příkazem. Příkaz `ls` vypíše strukturu [34].

```
[root@Server ~]# targetcli ls
o- / ..... [..]
  o- backstores ..... [..]
    | o- block ..... [Storage Objects: 0]
    | o- fileio ..... [Storage Objects: 0]
    | o- pscsi ..... [Storage Objects: 0]
    | o- ramdisk ..... [Storage Objects: 0]
    o- iscsi ..... [Targets: 0]
  o- loopback ..... [Targets: 0]
[root@Server ~]#
```

Obrázek 5 iSCSI targetcli [zdroj: vlastní]

V adresáři `backstores` se nachází čtyři možnosti uložení exportovaných LUN. První možností je export bloku. Blokové zařízení může reprezentovat celý disk, LVM disk nebo jiné blokové zařízení. Dále se také využívá export `fileio`, tedy export souboru, zbylé možnosti nejsou tak časté. Disk `sdb` bude sdílen právě pomocí `block`. Pomocí příkazu `/backstores/block create bcluna /dev/sdb` se vytvoří LUNa `bcluna`.

V rámci adresáře `iscsi` se poskytují konfigurace nutné k iSCSI exportu, například kterému klientovi (initiatoru) bude umožněno zobrazit vytvořené LUNy, jaké rozhraní bude použito ke sdílení dat na konkrétní lince a také musí být definován seznam povolených přístupů (ACL), aby se tato konfigurace dostala ke správným iniciátorům. Aby bylo možné iSCSI export nakonfigurovat je nutné ho inicializovat pomocí příkazu `/iscsi create`.

```
[root@Server ~]# targetcli
targetcli shell version 2.1.53
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/> /backstores/block create bcluna /dev/sdb
Created block storage object bcluna using /dev/sdb.
/> /iscsi create
Created target iqn.2003-01.org.linux-iscsi.server.x8664:sn.5b1bde136dc7.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
/> ls
o- / ..... [..]
  o- backstores ..... [..]
    | o- block ..... [Storage Objects: 1]
    |   o- bcluna ..... [/dev/sdb (8.0GiB) write-thru deactivated]
    |     o- alua ..... [ALUA Groups: 1]
    |       o- default_tg_pt_gp ..... [ALUA state: Active/optimized]
    | o- fileio ..... [Storage Objects: 0]
    | o- pscsi ..... [Storage Objects: 0]
    | o- ramdisk ..... [Storage Objects: 0]
    o- iscsi ..... [Targets: 1]
      o- iqn.2003-01.org.linux-iscsi.server.x8664:sn.5b1bde136dc7 ..... [TPGs: 1]
        o- tpg1 ..... [no-gen-acls, no-auth]
          o- acls ..... [ACLs: 0]
          o- luns ..... [LUNs: 0]
          o- portals ..... [Portals: 1]
          o- 0.0.0.0:3260 ..... [OK]
        o- loopback ..... [Targets: 0]

/>
* clearconfig / @last backstores/ iscsi/ loopback/ bookmarks cd
  saveconfig  exit  get  help  ls  pwd  refresh  restoreconfig
  saveconfig sessions set status version
```

Obrázek 6 Inicializace iSCSI export [zdroj: vlastní]

iSCSI export může shlukovat více adresářů, obsažených v target portal group (TPG). TPG je součástí každého iSCSI exportu a má tři podadresáře. Access control lists (ACLs) definuje seznam přístupů pro každého jednotlivého iSCSI klienta. Luns obsahuje seznam LUNů,

kteřé exportujeme tímto exportem. LUNs jsou definované v backstores a po úspěšném ověření klientů budou viditelné na klientech. Portals obsahuje seznam portálů neboli cest, kudy budou LUNy v tomto exportu exportovány. Pokud definujeme více než jeden portál (více než jednu cestu), klienti budou mít k dispozici více cest ke každému LUNu, což se da využít k multipathingu [34]. K přípravě LUN k exportu slouží příkaz:

```
[root@Server~]#/iscsi/iqn.2003-01.org.linux-iscsi.server.x8664:sn.776c62bc7e07/tpg1/luns create /backstores/block/bcluna
```

K nadeřinování ACL a klientů

```
[root@Server~]#/iscsi/iqn.2003-01.org.linux-iscsi.server.x8664:sn.776c62bc7e07/tpg1/acls create iqn.1994-05.com.redhat:b6f228ffd6b6
```

```
[root@Server~]#/iscsi/iqn.2003-01.org.linux-iscsi.server.x8664:sn.776c62bc7e07/tpg1/acls create iqn.1994-05.com.redhat:badb92f5e46e
```

```
> /backstores/block create bcluna /dev/sdb
Created block storage object bcluna using /dev/sdb.
> /iscsi create
Created target iqn.2003-01.org.linux-iscsi.server.x8664:sn.33152fe87e06.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
> /iscsi/iqn.2003-01.org.linux-iscsi.server.x8664:sn.297efe2a2111/tpg1/luns create /backstores/block/bcluna
No such path /iscsi/iqn.2003-01.org.linux-iscsi.server.x8664:sn.297efe2a2111
> clearconfig confirm=True
All configuration cleared
> /backstores/block create bcluna /dev/sdb
Created block storage object bcluna using /dev/sdb.
> /iscsi create
Created target iqn.2003-01.org.linux-iscsi.server.x8664:sn.34540cc051cd.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port 3260.
> /iscsi/iqn.2003-01.org.linux-iscsi.server.x8664:sn.34540cc051cd/tpg1/luns create /backstores/block/bcluna
Created LUN 0.
> ls
o- / ..... [..]
o- backstores ..... [..]
o- block ..... [Storage Objects: 1]
o- bcluna ..... [/dev/sdb (8.0GiB) write-thru activated]
o- alua ..... [ALUA Groups: 1]
o- default_tg_pt_gp ..... [ALUA state: Active/optimized]
o- fileio ..... [Storage Objects: 0]
o- pscsi ..... [Storage Objects: 0]
o- ramdisk ..... [Storage Objects: 0]
o- iscsi ..... [Targets: 1]
o- iqn.2003-01.org.linux-iscsi.server.x8664:sn.34540cc051cd ..... [TPGs: 1]
o- tpg1 ..... [no-gen-acls, no-auth]
o- acls ..... [ACLs: 0]
o- luns ..... [LUNS: 1]
o- lun0 ..... [block/bcluna (/dev/sdb) (default_tg_pt_gp)]
o- portals ..... [Portals: 1]
o- 0.0.0.0:3260 ..... [OK]
o- loopback ..... [Targets: 0]
>
```

Obrázek 7 Přidání LUN k exportu [zdroj: vlastní]

Targetcli ukládá deset záloh konfiguračních souborů do /etc/target/backup. Pokud se tedy nastavení z jakéhokoli důvodu smaže nebo se vyskytne chyba, neměl by být problém najít poslední vhodnou konfiguraci a nahradit ji stávající [34].

```

/> /iscsi/iqn.2003-01.org.linux-iscsi.server.x8664:sn.5b1bde136dc7/tpg1/luns create /backstores/block/bcluna
Created LUN 0.
/> /iscsi/iqn.2003-01.org.linux-iscsi.server.x8664:sn.5b1bde136dc7/tpg1/acls create iqn.1994-05.com.redhat:b6f228ffd6b6
Created Node ACL for iqn.1994-05.com.redhat:b6f228ffd6b6
Created mapped LUN 0.
/> /iscsi/iqn.2003-01.org.linux-iscsi.server.x8664:sn.5b1bde136dc7/tpg1/acls create iqn.1994-05.com.redhat:badb92f5e46e
Created Node ACL for iqn.1994-05.com.redhat:badb92f5e46e
Created mapped LUN 0.
/> ls
o- / ..... [..]
o- backstores ..... [..]
| o- block ..... [Storage Objects: 1]
| | o- bcluna ..... [/dev/sdb (8.0GiB) write-thru activated]
| | | o- alua ..... [ALUA Groups: 1]
| | | | o- default_tg_pt_gp ..... [ALUA state: Active/optimized]
| o- fileio ..... [Storage Objects: 0]
| o- pscsi ..... [Storage Objects: 0]
| o- ramdisk ..... [Storage Objects: 0]
o- iscsi ..... [Targets: 1]
| o- iqn.2003-01.org.linux-iscsi.server.x8664:sn.5b1bde136dc7 ..... [TPGs: 1]
| | o- tpg1 ..... [no-gen-acls, no-auth]
| | | o- acls ..... [ACLs: 2]
| | | | o- iqn.1994-05.com.redhat:b6f228ffd6b6 ..... [Mapped LUNs: 1]
| | | | | o- mapped_lun0 ..... [lun0 block/bcluna (rw)]
| | | | o- iqn.1994-05.com.redhat:badb92f5e46e ..... [Mapped LUNs: 1]
| | | | | o- mapped_lun0 ..... [lun0 block/bcluna (rw)]
| | | o- luns ..... [LUNs: 1]
| | | | o- lun0 ..... [block/bcluna (/dev/sdb) (default_tg_pt_gp)]
| | | o- portals ..... [Portals: 1]
| | | o- 0.0.0.0:3260 ..... [OK]
o- loopback ..... [Targets: 0]
/>

```

Obrázek 8 Hotová iSCSI server konfigurace [zdroj: vlastní]

4.1.1.1 Konfigurace iSCSI klientů

iSCSI klienti se připojují k targetu pomocí příkazů služby `iscsiadm`, která je poskytována balíkem `iscsi-initiator-utils`. K nakonfigurování je potřeba mít nastartované služby `iscsid` a `iscsi`. Dále stačí provést discovery LUNů na exportní IP. Zobrazí se nejen LUNy, které jsou namapovány v ACL pro tohoto hosta, ale i LUNy ostatních iSCSI exportů, pokud jsou definovány. Dalším příkazem už se host pokusí o přihlášení, a jestli bude zapsán v ACL a projde ověřením, tak se blok připojí na hosta [35].

```
iscsiadm -m discovery -t sendtargets -p 192.168.1.10
```

```
iscsiadm -m node -T iqn.2003-01.org.linux-iscsi.server.x8664:sn.776c62bc7e07 -p 192.168.1.10 -l
```

```

[root@Node01 ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1994-05.com.redhat:b6f228ffd6b6
[root@Node01 ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1994-05.com.redhat:b6f228ffd6b6
[root@Node01 ~]# █

[root@Node02 ~]# cat /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.1994-05.com.redhat:badb92f5e46e
[root@Node02 ~]# █

```

Obrázek 9 Příkaz k zobrazení InitiatorName [zdroj: vlastní]

Pomocí `lsblk` je možné zkontrolovat, zda se blokové jednotky k systému připojili. Také výpis příkazem `iscsiadm -m session -P 3` poskytne přehledný popis připojení.

```
[root@Node02 ~]# cat /proc/scsi/scsi
Attached devices:
Host: scsi1 Channel: 00 Id: 00 Lun: 00
  Vendor: VBOX      Model: CD-ROM          Rev: 1.0
  Type:   CD-ROM    ANSI SCSI revision: 05
Host: scsi3 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA       Model: VBOX HARDDISK   Rev: 1.0
  Type:   Direct-Access ANSI SCSI revision: 05
Host: scsi5 Channel: 00 Id: 00 Lun: 00
  Vendor: LIIO-ORG  Model: bcluna          Rev: 4.0
  Type:   Direct-Access ANSI SCSI revision: 05
[root@Node02 ~]# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                  8:0    0   20G  0 disk
├─sda1                8:1    0   1.9G  0 part /boot
├─sda2                8:2    0  18.1G  0 part 
│   └─almalinux-root 253:0    0   9.3G  0 lvm  /
│       └─almalinux-var 253:1    0   8.8G  0 lvm  /var
└─sdb                 8:16    0    8G  0 disk
sr0                 11:0    1 1024M  0 rom
```

Obrázek 10 Ověření připojení LUNy na uzel – 1. možnost [zdroj: vlastní]

```
[root@Node02 ~]# iscsiadm -m session -P 3
iSCSI Transport Class version 2.0-870
version 6.2.1.4-1
Target: iqn.2003-01.org.linux-iscsi.server.x8664:sn.6c6ebcf9f00 (non-flash)
Current Portal: 192.168.1.10:3260,1
Persistent Portal: 192.168.1.10:3260,1
*****
Interface:
*****
Iface Name: default
Iface Transport: tcp
Iface Initiatorname: iqn.1994-05.com.redhat:badb92f5e46e
Iface IPaddress: 192.168.1.12
Iface HWaddress: default
Iface Netdev: default
SID: 1
iSCSI Connection State: LOGGED IN
iSCSI Session State: LOGGED_IN
Internal iscsid Session State: NO CHANGE
*****
Timeouts:
*****
Recovery Timeout: 120
Target Reset Timeout: 30
LUN Reset Timeout: 30
Abort Timeout: 15
*****
CHAP:
*****
username: <empty>
password: *****
username_in: <empty>
password_in: *****
*****
Negotiated iSCSI params:
*****
HeaderDigest: None
DataDigest: None
MaxRecvDataSegmentLength: 262144
MaxXmitDataSegmentLength: 262144
FirstBurstLength: 65536
MaxBurstLength: 262144
ImmediateData: Yes
InitialR2T: Yes
MaxOutstandingR2T: 1
*****
Attached SCSI devices:
*****
Host Number: 5 State: running
scsi5 Channel 00 Id 0 Lun: 0
Attached scsi disk sdb State: running
```

Obrázek 11 Ověření připojení LUNy na uzel – 2. možnost [zdroj: vlastní]

4.1.2 Použití HA-LVM

Veškerá sdílená data budou na svazcích LVM, protože právě tímto způsobem budou klastru poskytnuty potřebné prostředky pro správu přístupu k datům. Pro aktivní/pasivní klastr bude stačit LVM dostupné při instalaci systému. V aktivním/pasivním klastru je LV, na kterém

jsou uložena sdílená data, aktivováno vždy jen na jednom uzlu. Pro správu sdílených prostředků aktivním/pasivním způsobem s pouze jedním aktivním uzlem, který potřebuje přístup k danému svazku LVM najednou, lze použít HA-LVM bez zamykací služby `lvmlckd` [36].

Většina aplikací může mít větší výkon v aktivní/pasivní konfiguraci, protože nejsou navrženy ani optimalizovány tak, aby běžely souběžně s jinými instancemi. Volba spuštění aplikace, která nepodporuje klastr na sdílených logických svazcích, může vést ke snížení výkonu. Důvodem je, že v těchto případech existuje režie klastrové komunikace pro samotné logické svazky, která může na běh aplikace působit spíše negativně, nežli pozitivně. Aby se aplikaci vyplatilo působit v klastru, musí být aplikace schopna dosáhnout zvýšení výkonu při práci se souborovými systémy klastru a logickými svazky s podporou klastru. Toho lze u některých aplikací a pracovních zátěží dosáhnout snadněji než u jiných. Určit, jaké jsou požadavky klastru a zda se dodatečné úsilí o optimalizaci pro provedení aktivního/aktivního klastru vyplatí, může mnohdy být nezodpovězená otázka [36].

4.2 DRBD

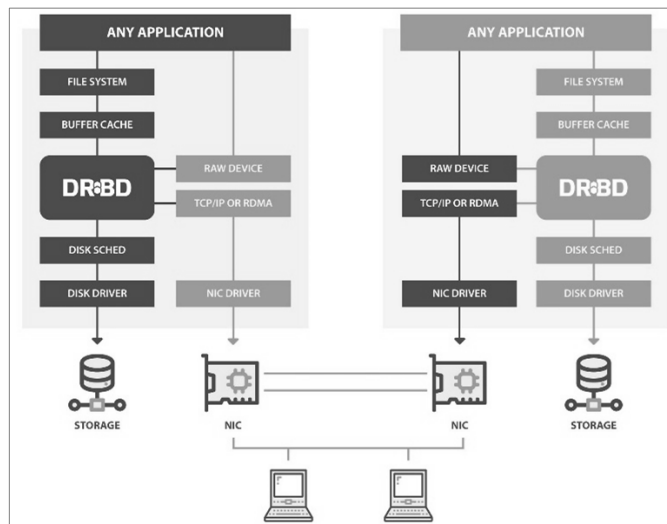
Distributed Replicated Block Device (DRBD) je softwarové, sdílené, replikované úložiště, které zrcadlí obsah blokových zařízení (pevné disky, oddíly, logické svazky atd.) mezi hostiteli.

DRBD zrcadlí data

- V reálném čase. Replikace probíhá nepřetržitě, zatímco aplikace upravují data v zařízení.
- Transparentně. Aplikace nemusí vědět, že data jsou uložena na více hostitelích.
- Synchronně nebo asynchronně. Při synchronním zrcadlení jsou aplikace informovány o dokončení zápisu poté, co byly zápisy provedeny na všech (připojených) hostitelích. Při asynchronním zrcadlení jsou aplikace upozorněny na dokončení zápisu, když jsou zápisy dokončeny lokálně, což je obvykle dříve, než se rozšíří na ostatní hostitele [37].

Základní funkčnost DRBD je implementována prostřednictvím modulu jádra Linuxu. Konkrétně DRBD představuje ovladač pro virtuální blokové zařízení a hierarchicky se nachází těsně u spodní části systémového I/O zásobníku. Proto je DRBD extrémně flexibilní a všestranné, což z něj dělá replikační řešení vhodné pro přidání vysoké dostupnosti téměř

jakékoli aplikace. DRBD nemůže měnit ani přidat vlastnosti do horních vrstev, například nemůže automaticky detekovat poškození souborového systému nebo přidat schopnost aktivního/aktivního klastrování do souborových systémů [37].



Obrázek 12 Pozice DRBD v rámci linuxového I/O zásobníku [39,
<https://linbit.com/wp-content/uploads/2020/03/DRBD-Diagram.jpg>]

DRBD lze nainstalovat pomocí balíků dodávané sponzorskou společností LINBIT. Balíky jsou dostupné například prostřednictvím repositářů. Balíky z těchto zdrojů jsou považovány za oficiální sestavení a jsou k dispozici pro tyto distribuce: RHEL, SUSE Linux Enterprise Server (SLES), Debian GNU/Linux a Ubuntu Server Edition.

4.2.1 Nástroje pro správu

4.2.1.1 Drbdadm

Nástroj pro správu sady programů DRBD-utils. Získává všechny konfigurační parametry z konfiguračního souboru umístěném v `/etc/drbd.conf` a funguje jako front-end pro `drbdsetup` a `drbdmeta`. `Drbdadm` nabízí režim „běhu na sucho“, vyvolaný `-d` volbou, který ukazuje, jaké odpovědi by `drbdsetup` a `drbdmeta` vracely, aniž by `drbdadm` skutečně volal tyto příkazy [38].

4.2.1.2 Drbdsetup

Konfiguruje modul DRBD, který byl načten do jádra. Všechny parametry pro `drbdsetup` musí být předány na příkazovém řádku. Oddělení `drbdsetup` od `drbdadm` umožňuje

maximální flexibilitu. Většina uživatelů bude jen zřídka potřebovat použít drbdsetup napřímo [38].

4.2.1.3 Drbdmeta

Umožňuje vytvářet, vypisovat, obnovovat a upravovat struktury metadat DRBD. Stejně jako drbdsetup, většina uživatelů bude muset jen zřídka používat přímo drbdmeta [38].

4.2.2 Role zdrojů

V DRBD každý zdroj má svou roli, která je buď primární, nebo sekundární [37].

- Zařízení v primární roli lze neomezeně používat pro operace čtení a zápisu. Může být použito pro vytváření a připojení souborových systémů, pro zápis nebo čtení blokových zařízení atd.
- Zařízení v sekundární roli přijímá všechny aktualizace z primárního zařízení, ale jinak zcela zakáže přístup. Nelze jej používat aplikacemi, a to ani pro čtení, ani pro přístup k zápisu. Důvodem zákazu je nutnost zachování koherence mezi-paměti, což by nebylo možné, pokud by byl jakkoli zpřístupněn sekundární zdroj.

Roli zdroje lze samozřejmě změnit buď ručním zásahem pomocí nějakého automatizovaného algoritmu aplikací pro správu klastru nebo automaticky. Změna role zdroje ze sekundární na primární se nazývá povýšení, zatímco obrácená operace se nazývá degradace [39].

V DRBD může být zdroj v primární roli na dvou uzlech současně. V režimu jednoho primárního je prostředek v každém okamžiku v primární roli pouze u jednoho člena klastru. Protože je zaručeno, že s daty v každém okamžiku manipuluje pouze jeden uzel klastru, lze tento režim použít s jakýmkoli konvenčním souborovým systémem (ext3, ext4, XFS, atd.) [39].

Pokud je použit sdílený souborový systém klastru, který využívá distribuovaného správce zámku, jako například GFS nebo OCFS2, je možné použít režim souběžného přístupu k datům. Tento režim nasazení DRBD se jmenuje dual-primary a je preferovaný pro klastry vyvažující zátěž [39].

DRBD se dá jednoduše použít také v Pacemaker klastru. Použití DRBD ve spojení s klastrem Pacemaker je pravděpodobně nejčastější případ použití DRBD. Pacemaker dělá z DRBD extrémně výkonnou aplikaci v široké škále scénářů použití. Pacemaker se může

starat i o roli zdroje, kterou může přepínat automaticky. K použití DRBD replikovaného úložiště, které se chová a vypadá jako místní úložiště, je potřeba integrace do klastru Pacemakeru, která se provádí nasměrováním přípojných bodů zařízení do prostředků klastru [39].

Nejprve se musí použít funkce *auto-promote*, která zpřístupní, že se v případě potřeby automaticky nastaví vhodný uzel jako primární.

V konfiguračním souboru je potřeba přidat *common { options { auto-promote yes; }}*

Nyní je potřeba nasměrovat přípojný bod úložiště do klastru, např. prostřednictvím souborového systému. Příklad příkazu:

```
crm configure primitive fs_test ocf:heartbeat:Filesystem params
device="/dev/drbd/drbd1" directory="/var/log/html" fstype="ext4".
```

Agent prostředku *ocf:linbit:drbd* OCF poskytuje také funkci *master/slave*, což umožňuje Pacemakeru spouštět a monitorovat prostředek DRBD na více uzlech a podle potřeby povyšovat a degradovat [39].

V tomto řešení ale nastaly problémy. DRBD úložiště mělo sloužit k replikování dat mezi uzly. Konfigurace a použití už je dlouhou dobu ověřená a k replikování dat mezi dvěma disky by nemělo nic bránit. Ovšem v tomto případě byl nasdílen na oba uzly jeden disk. Ten disk ať už je připojen k jednomu nebo druhému uzlu vždy obsahuje stejná data. Když tedy systému přišel pokus o vytvoření LVM nebo adresáře na obou uzlech, na druhém to zahlásilo například „*A volume group called drbdplace already exists*“. Použití DRBD v tomto případě tedy nemá smysl, jelikož oba uzly zapisují střídavě na jeden disk, který si vždy drží aktuální data a informace a ta jsou dostupná právě aktivnímu uzlu.

4.3 Instalace a konfigurace uzlů klastru

Po instalaci systému a nakonfigurování sítě nebyl aktivován HA repositář. V souboru *almalinux-ha.repo* umístěném v */etc/yum.repos.d/* je potřeba nastavit *enabled* na hodnotu 1, pro jistotu u všech tří balíků, které konfigurační soubor obsahuje.

Po povolení HA repositáře je možné stáhnout balíky *pcs* a *pacemaker*, ze kterých se sestaví klastr. Klastr se konfiguruje například v souboru *corosync.conf* v */etc/corosync/*. Jak již bylo zmíněno *corosync* je program, který slouží k základním potřebám členství a komunikace mezi uzly. Takže v *corosync.conf* je možné nastavit transport a komunikaci klastru, seznam uzlů a přístup k nim, quorum nebo například kam má klastr logovat informace.

K sestavení klastru je potřeba nastavit heslo na každém uzlu pro uživatele hacluster, což je účet pro správu klastru, dále povolit a nastartovat službu pcsd.service. Ověření uživatele hacluster pro každý uzel provede příkaz `[root@Node01 ~]#pcs host auth Node01 Node02`, který vybídne k přihlášení a autorizaci uzlů. Zde se zadává heslo pro uživatele hacluster, které je potřeba pro přihlášení do webového rozhraní klastru [kapitola 4.4]. Po ověření už stačí jen klastr sestavit pomocí příkazu `[root@Node01 ~]#pcs cluster setup bc_cluster -start node01 node02`. Příkazem `[root@Node01 ~]#pcs cluster start -all` poté klastr nastartuje a již je možné ho monitorovat, například příkazem `crm_mon`. Takto vytvořený klastr ale nemá žádné prostředky, tedy nic nespravuje, nic neřídí. Konfigurace těchto prostředků pomocí služby pcs může být náročná [40].

Pro zjednodušení konfigurace, správu a řešení problémů bylo doinstalováno rozhraní crmsh. Crmsh je součástí projektu Clusterlabs, jedná se o rozhraní příkazového řádku pro správu klastru s vysokou dostupností na systémech GNU/Linux. Zjednodušuje řešení problémů klastrů založených na pacemakeru tím, že poskytuje výkonnou a intuitivní sadu funkcí [41].

Crmsh může fungovat jednak jako interaktivní shell s doplňováním příkazů pomocí tabulátoru a vloženou dokumentací, tak jako nástroj příkazového řádku. Může být také použit v dávkovém režimu k provádění příkazů ze souborů [41].

V této práci se využívá nápověda, kterou crmsh poskytuje. Příkazy v crm se mohou psát postupně, s nápovědou ke každému kroku, nebo jako jeden příkaz. Pro vypsání seznamu možných agentů prostředků (Resource Agents), slouží příkaz `crm ra classes`. A pro vypsání prostředků, které tento agent poskytuje, slouží příkaz `crm ra list <class>`.

```
[root@Node02 ~]# crm
crm(live/Node02)# ra
crm(live/Node02)ra#
cd      classes      help      info      list      ls      providers      quit      up      validate
crm(live/Node02)ra# classes
lsb
ocf / heartbeat linbit openstack pacemaker
service
stonith
systemd
crm(live/Node02)ra# list ocf
CTDB          ClusterMon    Delay         Dummy         Filesystem    HealthCPU     HealthIOWait
HealthSMART   IPaddr2      IPaddr2      IPscaddr      LVM-activate  MailTo        NodeUtilization
NovaEvacuate  Route        SendArp      Squid          Stateful      SysInfo       SystemHealth
VirtualDomain Xinetd       apache       contribute    aws-vpc-move-ip controlld      crypt         awsip
awsvip        azure-events azure-lb      conntrackd    galera        garbd         db2
dhcpd         docker       ethmonitor   exportfs       nova-compute-wait mysql          iSCSILogicalUnit
iSCSITarget   iface-vlan   ifspeed      lvmlockd      oraasm        nagios        named
nfsnotify     nfsserver    ifspeed      nova-compute-wait oraasm        nagios        oralsnr
pgsql         ping         podman       portblock     postfix       rabbitmq-cluster redis
remote        rsyncd       slapd        storage-mon    sybaseASE     symlink       tomcat
vdo-vol
crm(live/Node02)ra#
```

Obrázek 13 Výpis Resource Agents [zdroj: 41,42]

Klastr je tímto připraven na přidání prostředků.

4.3.1 Konfigurace HA-LVM v klastru

Jelikož disk je sdílený přes iSCSI, stačí vytvořit logický svazek LVM na jednom nodu. Nejdříve je potřeba ověřit, že sdílený disk `/dev/sdb` je připojen k uzlu. Po ověření se příkazem `fdisk /dev/sdb` spustí program `fdisk`, který slouží k vytváření diskových oddílů. Oddíl `/dev/sdb1` byl nakonfigurován jako primární s Partition number: 1 a defaultním rozsahem oddílu, pomocí příkazu `w` se provedené změny zapíše do tabulky, která se uloží do Master Boot Record. Příkazem `p` se vypíše tabulka oddílů.

```
Command (m for help): p
Disk /dev/sdb: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 33550336 bytes
Disklabel type: dos
Disk identifier: 0x930f1854

Device            Boot Start      End  Sectors  Size Id Type
/dev/sdb1          65528 16777215 16711688   8G 83 Linux
```

Obrázek 14 výstup příkazu `fdisk` – list partition table [zdroj: `fdisk`]

Následně se pomocí příkazů:

- `[root@Node01 ~]#pvcreate /dev/sdb1`
- `[root@Node01 ~]#vgcreate bc_vg /dev/sdb1`
- `[root@Node01 ~]#lvcreate -l 100%FREE -n bc_lv bc_vg`
- `[root@Node01 ~]#mkfs.ext4 /dev/bc_vg/bc_lv`
- `[root@Node01 ~]#mount /dev/bc_vg/bc_lv /var/www`

vytvoří logický svazek LVM a poté na tomto svazku vytvoří souborový systém `ext4`. Posledním příkazem se připojí LVM oddíl na adresář `/var/www` a nahradí stávající obsah.

```
[root@Node01 ~]# mount | grep ext4
/dev/mapper/bc_vg-bc_lv on /var/www type ext4 (rw,relatime,stripe=8191)
```

Obrázek 15 Ověření připojení oddílu, zdroj: vlastní

Pomocí příkazu `[root@Node01 ~]#crm ra info LVM-activate` byly vypsány potřebné parametry pro konfiguraci prostředku LVM v klastru. Pro tento prostředek byl zvolen exkluzivní přístup tak, aby přístup k datům byl umožněn pouze jednomu, a to aktivnímu uzlu. Podle dokumentace se tato schopnost aktivuje parametrem `vg_access_mode=system_id` a tak byl také prostředek nakonfigurován. Podobným způsobem byly nakonfigurovány i zbylé parametry a ostatní prostředky klastru.

```

activation_mode (string, [exclusive]): Logical volume activation mode
  The activation mode decides the visibility of logical volumes in the cluster. There
  are two different modes: "shared" for cluster filesystem and "exclusive" for local
  filesystem. With "shared", an LV can be activated concurrently from multiple nodes.
  With "exclusive", an LV can be activated by one node at a time.

  This option only has effect on "lvmlockd"/"clvmd" vg_access_mode. For "system_id"
  and "tagging", they always mean exclusive activation.

tag (string, [pacemaker]): The tag used for tagging activation mode
  The tag used for tagging activation mode.

```

Obrázek 16 Ústřížek výpisu crm ra info LVM-activate [zdroj: 41,42]

Celková konfigurace prostředku, který byl pojmenován my_lvm vypadá následovně:

```

[root@Node01 ~]# pcs resource config my_lvm
Resource: my_lvm (class=ocf provider=heartbeat type=LVM-activate)
Attributes: vg_access_mode=system_id vgname=bc_vg
Operations: monitor interval=30s timeout=90s (my_lvm-monitor-interval-30s)
            start interval=0s timeout=90s (my_lvm-start-interval-0s)
            stop interval=0s timeout=90s (my_lvm-stop-interval-0s)

```

Obrázek 17 Prostředek klastru – LVM [zdroj: vlastní]

4.3.2 Konfigurace Filesystem v klastru

Prostředek OCF:heartbeat:Filesystem spravuje souborový systém na sdíleném oddílu. Skripty tohoto prostředku monitorují, zda je souborový systém připojen a snaží se ho připojit na aktivní uzel. Z manuálové stránky tohoto prostředku je patrné, že nutné jsou pouze tři parametry a to, které zařízení bude připojovat, kam bude zařízení připojeno a typ souborového systému.

```

Parameters (*: required, []: default):
device* (string): block device
  The name of block device for the filesystem, or -U, -L options for mount, or NFS mount specification.
directory* (string): mount point
  The mount point for the filesystem.
fstype* (string): filesystem type
  The type of filesystem to be mounted.

```

Obrázek 18 Ústřížek výpisu crm ra info Filesystem [zdroj: 41,42]

Finální konfigurace prostředku, který byl pojmenován my_fs vypadá následovně:

```

[root@Node01 ~]# pcs resource config my_fs
Resource: my_fs (class=ocf provider=heartbeat type=Filesystem)
Attributes: device=/dev/bc_vg/bc_lv directory=/var/www fstype=ext4

```

Obrázek 19 Prostředek klastru – Filesystem [zdroj: vlastní]

4.3.3 Aplikace

Pro ověření funkčnosti HA klastru byl na aplikačních uzlech spuštěn webserver. Inspirací pro tvorbu webserveru byl web jakpsatweb.cz a článek <https://www.jakpsatweb.cz/php/jak-zacit.html>. Aplikace byla pro jednoduchost nejprve napsána a zprovozněna v prostředí Windows 10 bez HA klastru a poté předělána pro aplikační uzly klastru na OS Almalinux 8.5.

Úkolem webového serveru (webserveru) je poskytovat webové stránky po požadavku oprávněných klientů na internetu. K dosažení tohoto cíle funguje jako prostředník mezi serverem a klientskými stroji. Databázové servery, poštovní servery, webové servery a další používají různé druhy serverového softwaru a každá z těchto aplikací může přistupovat k určitým souborům uloženým na fyzickém serveru a webserver jim pomáhá obsluhovat a používat tyto soubory k různým účelům [43,44].

Webserver v této práci se skládá ze tří hlavních součástí:

- Server Apache
- Modul PHP pro Apache
- Databáze MariaDB

4.3.3.1 *Server Apache*

Apache je software, který běží na serveru zapojeném do internetové sítě. Jeho úkolem je navázat spojení mezi serverem a prohlížeči návštěvníků webových stránek a zároveň mezi nimi doručovat soubory tam a zpět. Software Apache je kompatibilní s jakýmkoli operačním systémem. Když chce návštěvník načíst stránku na webu, jeho prohlížeč odešle požadavek na server a Apache vrátí odpověď se všemi požadovanými soubory (text, obrázky, atd...). Server a klient komunikují prostřednictvím protokolu http a Apache je zodpovědný za hladkou a bezpečnou komunikaci mezi nimi. Apache navíc poskytuje spoustu modulů, které správci serveru umožňují zapínat a vypínat zajímavé funkce. Webserver Apache má moduly pro zabezpečení, ukládání do mezipaměti, přepisování URL, ověřování heslem a spoustu dalších [44, 45].

4.3.3.2 *PHP*

PHP je zkratka pro Hypertext Preprocessor. Je to skriptovací programovací jazyk, který je široce používán pro vývoj webových aplikací. Používá se pro skriptování na straně serveru a je open source. Oblíbeným se stal především díky jednoduchosti použití a spoustě funkcí,

které nabízí. Zjednodušeně se PHP web skládá z kódu HTML, CSS, Javascriptu a kódu PHP. PHP kód se spustí na straně serveru, ale výsledek zobrazený prohlížečem klientů je přijat ve formátu HTML. PHP kód se vkládá do obyčejných HTML souborů mezi značky `<?php` a `>` a při požadavku na PHP stránku server prochází soubor a PHP kódy zpracuje pomocí skriptu, vyhodnotí je a klientovi posílá už čistou HTML odpověď [45,46].

Jelikož modul PHP pro Apache je již součástí balíku php není potřeba jej samostatně instalovat. [zdroj: <https://access.redhat.com/discussions/6395021>] Pro databázi byl však doinstalován balík php-mysqld.

4.3.3.3 Databáze MariaDB

MariaDB je relační databáze, která je komunitou vyvíjeným forkem MySQL. Snahou projektu MariaDB je udržení licence svobodného softwaru GNU GPL pro MySQL databázi, o kterou pomalu přichází pod vedením firmy Oracle. Když Oracle koupil Sun Microsystems, objevila se spousta otázek kolem open source projektů, které pod Sun patřily. Jedním z těchto produktů byla i databáze MySQL. Na této databázi stojí ohromné množství projektů. Oracle koupil databázi MySQL společně s firmou Sun a od té doby postupně uzavírá její vývoj. Iniciativa, díky které tento fork vzniká, pochází od původních vývojářů MySQL, kteří se obávali o další směřování tohoto softwaru po jeho odkoupení společností Oracle [47,48].

Hlavním vývojářem je Michael „Monty“ Widenius, který je původním zakladatelem MySQL a Monty Program AB. Michael opustil Sun z důvodu nenaplnění jeho představ o vývoji MySQL a vytvořil novou společnost The Monty Program. Ta začala pracovat na forku MySQL databáze s názvem MariaDB. MariaDB si postupně začala budovat jméno a zlom nastal v roce 2012, kdy byla založena organizace MariaDB Foundation. Zakladateli byl už představený Michael Widenius a s ním David Axmark a Allan Larsson. Všichni byly také zakladateli původní společnosti MySQL AB [47,49].

4.3.3.4 Instalace aplikace

Konfigurace webového serveru byla otestována v jiném prostředí a výsledná konfigurace byla přenesena na virtuální stroj Node01, kde byla aplikace zprovozněna nejprve mimo klastr a poté přidána mezi klastrové prostředky.

Instalace na Node01 byla zahájena nainstalováním balíku httpd a php. Následně byl vytvořen adresář html a mysql na sdíleném LVM svazku /var/www/ tak, aby byla rozdělena konfigurace a úložiště pro Apache a MariaDB databázi. Do adresáře /var/www/html byly

nakopírovány obrázky a konfigurace webu, která byla otestována v prostředí Windows. Po spuštění httpd příkazem `systemctl start httpd`, je již možné nahlédnout v prohlížeči na defaultní stránku Apache. Jestliže je PHP nainstalováno a mělo dostatečné práva na vytvoření souboru `php.conf` v cestě `/etc/httpd/conf.d/` zobrazí se v prohlížeči informace o nainstalované verzi PHP.

vojtuvweb.cz	
PHP Version 7.2.24 	
System	Linux Node01 4.18.0-348.12.2.el8_5.x86_64 #1 SMP Wed Jan 19 14:35:04 EST 2022 x86_64
Build Date	Oct 22 2019 08:28:36
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-dom.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gd.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-intl.ini, /etc/php.d/20-json.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-mysqli.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-mysqli.ini, /etc/php.d/30-pdo_mysql.ini, /etc/php.d/30-pdo_sqlite.ini, /etc/php.d/30-wddx.ini, /etc/php.d/30-xmlreader.ini, /etc/php.d/40-zip.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, compress.bzip2, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, bzip2.*, convert.iconv.*

Obrázek 20 Home page webu [zdroj: vlastní]

Po přidání souboru `web.conf` do adresáře `/etc/httpd/conf.d/` bude možné přistupovat z prohlížeče do adresáře `/var/www/html`, kde je nakonfigurován web a zobrazit jej vykreslený pro klienta. V adresáři `/var/www/html/hello` je soubor `about.php`, který slouží pro zápis uživatelů, kteří podají svůj kontakt na webu, do MariaDB databáze, soubor `db-connect-test.php`, který slouží pro testování spojení PHP s databází, soubor `hello.php`, který obsahuje jednu z nejzákladnějších konfigurací PHP webu, díky které je testováno spojení Apache s PHP, a to vytištění „Hello, World!“ v prohlížeči a soubor `testik.php`, ve kterém je obsah webové stránky.

Na tento uzel byly navíc doinstalovány balíky `mariadb-server` a `php-mysqldb` spolu se závislostmi. Adresář `/var/lib/mysql` byl přesunut na sdílené LVM do adresáře `/var/www/mysql/` a adresář `/var/lib/mysql` byl pomocí příkazu `ln -s /var/www/mysql`

/var/lib/ symbolicky adresován na /var/www/mysql/ adresář. Díky tomuto symbolickému odkazu, který pracuje jako link na sdílené úložiště, je zajištěno, že každý aktivní uzel bude mít stejná data a stejnou konfiguraci databáze. V konfiguraci prostředku MariaDB v klastru totiž bude nastaveno, ať si aktivní uzel data tahá právě z tohoto adresáře.

```
[root@Node01 html]# ll /var/lib/mysql  
lrwxrwxrwx 1 root root 15 Apr 25 16:01 /var/lib/mysql -> /var/www/mysql/
```

Obrázek 21 Symbolický odkaz adresářů [zdroj: vlastní]

Na druhý uzel, který byl doposud nečinný, byly nainstalovány balíky a vytvořen symbolický odkaz mezi adresáři stejně jako na prvním uzlu.

4.3.3.5 Přidání prostředků aplikace do klastru

Uzly klastru již tedy jsou připraveny pro přidání aplikace do klastru. Prvními prostředky, které se do klastru dále přidají, jsou VIP adresa a služba SendARP. Konfigurace těchto prostředků je velice snadná, je třeba být ale opatrný při výběru adresy, aby nedošlo k tomu, že použijeme adresu v síti, která je již používána.

```
[root@Node01 html]# pcs resource config vip_web  
Resource: vip_web (class=ocf provider=heartbeat type=IPaddr2)  
Attributes: cidr_netmask=28 ip=192.168.1.14 nic=enp0s8:1  
[root@Node01 html]# pcs resource config vip_web_arp  
Resource: vip_web_arp (class=ocf provider=heartbeat type=SendArp)  
Attributes: ip=192.168.1.14 nic=enp0s8:1
```

Obrázek 22 Prostředek klastru – VIP a SendARP [zdroj: vlastní]

Jelikož není potřeba, aby Apache běžel na obou uzlech zároveň, může být také přidán do klastru a spravován Pacemakerem pouze na aktivním uzlu. Když by Apache běžel na obou uzlech zároveň, nemělo by to vliv na dostupnost webu, jelikož sdílené úložiště a VIP adresu, tedy tu, na kterou se budou tázat klienti, bude hostovat vždy jen aktivní uzel a na druhém uzlu by tedy Apache běžel zbytečně.

```
[root@Node01 html]# less /var/pcs resource config httpd  
Resource: httpd (class=ocf provider=heartbeat type=apache)  
Attributes: configfile=/etc/httpd/conf/httpd.conf statusurl=http://127.0.0.1/server-status  
Operations: start interval=0s timeout=40s (httpd-start-0s)  
             stop interval=0s timeout=60s (httpd-stop-0s)  
             monitor interval=20s timeout=20s (httpd-monitor-20s)
```

Obrázek 23 Prostředek klastru – Apache [zdroj: vlastní]

Posledním prostředkem klastru je MariaDB databáze. Pro tento prostředek má Pacemaker klastr agenta prostředků ocf:heartbeat:mysql, který spravuje instanci MySQL databáze. Parametry tohoto prostředku v této práci jsou nastaveny téměř defaultně. Důležité je

v konfiguraci nastavení parametru `datadir="/var/lib/mysql/"`, takhle byla totiž tvořena a realizována myšlenka sdíleného úložiště se symbolickým odkazem na tento adresář.

```
[root@Node02 ~]# pcs resource config mysql mysql
Resource: mysql_mysql (class=ocf provider=heartbeat type=mysql)
Attributes: binary=/usr/bin/mysqld_safe config=/etc/my.cnf datadir=/var/lib/mysql/ group=mysql log=/var/log/mariadb/mariadb.log
pid=/run/mariadb/mariadb.pid socket=/var/lib/mysql/mysql.sock user=mysql
Meta Attrs: target-role=Started
Operations: monitor interval=30s timeout=30s (mysql_mysql-monitor-30s)
            start interval=0 timeout=120 (mysql_mysql-start-0)
            stop interval=0 timeout=120 (mysql_mysql-stop-0)
```

Obrázek 24 Prostředek klastru – Mysql [zdroj: vlastní]

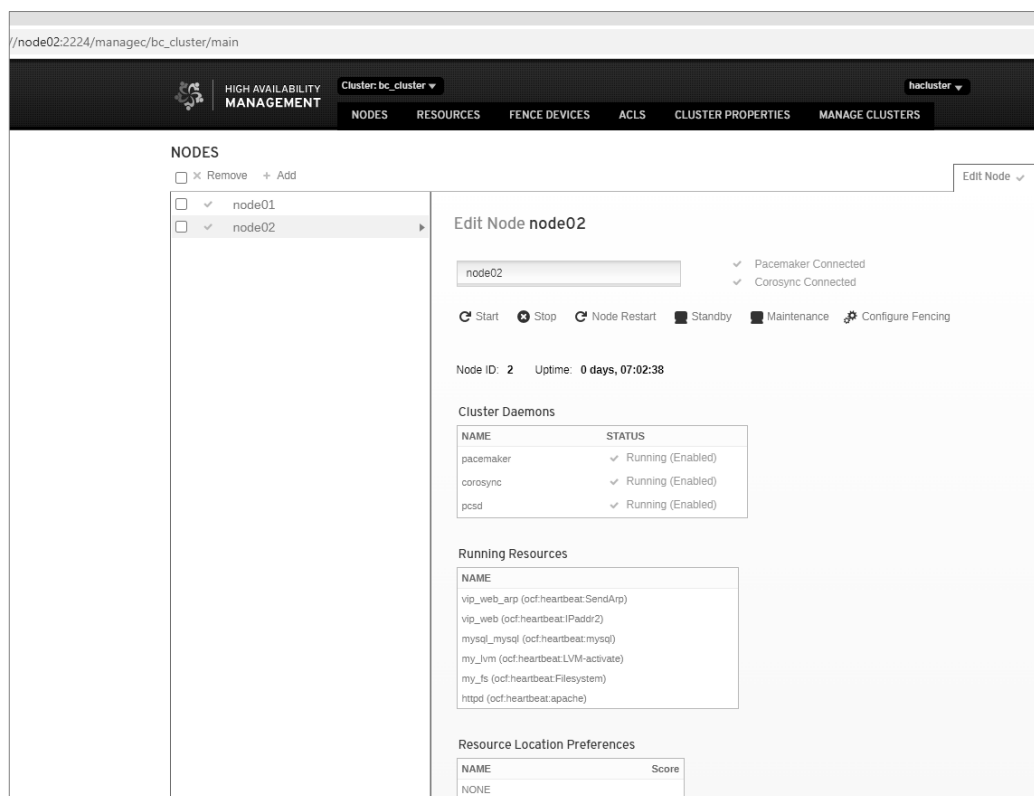
Finální konfigurace prostředků klastru tedy vypadá následovně:

```
[root@Node01 ~]# crm configure show
node 1: node01 \
  attributes standby=off
node 2: node02 \
  attributes standby=off
primitive httpd apache \
  params configfile="/etc/httpd/conf/httpd.conf" statusurl="http://127.0.0.1/server-status" \
  op start interval=0s timeout=40s \
  op stop interval=0s timeout=60s \
  op monitor interval=20s timeout=20s
primitive my_fs Filesystem \
  params device="/dev/bc_vg/bc_lv" directory="/var/www" fstype=ext4
primitive my_lvm LVM-activate \
  params vg_access_mode=system_id vgname=bc_vg \
  op monitor interval=30s timeout=90s \
  op start interval=0s timeout=90s \
  op stop interval=0s timeout=90s
primitive mysql_mysql mysql \
  params binary="/usr/bin/mysqld_safe" config="/etc/my.cnf" user=mysql group=mysql log="/var/log/mariadb/mariadb.log"
pid="/run/mariadb/mariadb.pid" datadir="/var/lib/mysql/" socket="/var/lib/mysql/mysql.sock" \
  op monitor interval=30s timeout=30s \
  op start interval=0 timeout=120 \
  op stop interval=0 timeout=120 \
  meta target-role=Started
primitive vip_web IPAddr2 \
  params ip=192.168.1.14 cidr_netmask=28 nic="enp0s8:1"
primitive vip_web_arp SendArp \
  params ip=192.168.1.14 nic="enp0s8:1"
```

Obrázek 25 Prostředky klastru [zdroj: vlastní]

4.4 Konfigurace klastru na Webu

Pacemaker klastr je možné konfigurovat také z prohlížeče. Při spuštění služby `pcsd.service` začne na portu 2224 poslouchat webové rozhraní klastru. Přihlášení do této správy je pod uživatelem `hacluster`.



Obrázek 26 Management okno klastru ve webovém prohlížeči [zdroj: vlastní]

Na stránce NODES je výpis uzlů vybraného klastru a je zde možnost zastavit, spustit, restartovat, odstavit nebo nastavit na uzlu status údržby. Také jsou zde zobrazení démoni klastru a aktivní prostředky. Na stránce RESOURCES jsou prostředky klastru, je možné je povolit, zakázat, vyčistit, restartovat nebo spravovat. Další důležitou stránkou je CLUSTER PROPERTIES, zde je vypsán souhrn vlastností a možností klastru. Tabulka na stránce CLUSTER PROPERTIES shrnuje vlastnosti klastru a ukazuje výchozí hodnoty vlastností a možné hodnoty, které lze pro tyto vlastnosti nastavit. Na stránce MANAGE CLUSTERS je možné odebrat, přidat existující, zničit nebo vytvořit nový klastr.

4.5 Experimentální testy

Na fyzických strojích dochází k plánovaným či neplánovaným odstávkám. Při takovéto odstávce je nutné virtuální stroje, které běží na fyzickém stroji, odstavit z provozu a vypnout. Příkladem může být aktualizace jádra Linuxu, fyzické přidání RAM paměti nebo CPU do stroje nebo jen plánovaný klastr failover, který má za úkol prověřit, že je infrastruktura dobře připravená a nedojde k velkému výpadku služeb. K aktualizaci jádra na virtuálním stroji je potřeba také uzel odstavit a po aktualizaci balíčků uzel restartovat, aby se načel systém na zaktualizovaném jádře. Klastr by tohle měl umožnit s minimálním výpadkem.

4.5.1 Výpadek při odstavení klastru na uzlu

Výpadek služeb byl měřen skriptem `job.sh`, který je v příloze a porovnáván bude s logem `pacemaker`, který se nachází na virtuálních strojích v cestě `/var/log/pacemaker/`. Skript byl spuštěn tak, aby se 30 sekund dotazoval na konkrétní obsah na webserveru. Agent VIP adresy byl nastaven v klastru, ať se spustí vždy jako poslední služba, tedy vždy až je zapnutá databáze připojená na Apache webserver. Tímto bylo zajištěno, že v době kdy přijde korektní odpověď na dotaz z VIP adresy, dokončil klaster už přepnutí. Odstavení jednoho uzlu bylo provedeno v příkazovém řádku na druhém uzlu, který byl zrovna aktivní a hostoval služby. Test začal spuštěním skriptu a následným příkazem `crm node standby`, dostal `pacemaker` povel odstavit uzel a přepnout služby na druhý.

Skript zachytil výpadek v čase 11:32:27 a správnou odpověď dostal v čase 11:32:38, tedy přibližně 11 sekund byly služby nedostupné na webserveru.

```
Oznaceni 1 OK Sun May 8 11:32:24 CEST 2022
Oznaceni 2 OK Sun May 8 11:32:25 CEST 2022
Oznaceni 3 NO Sun May 8 11:32:27 CEST 2022
Oznaceni 4 NO Sun May 8 11:32:28 CEST 2022
Oznaceni 5 NO Sun May 8 11:32:29 CEST 2022
Oznaceni 6 NO Sun May 8 11:32:30 CEST 2022
Oznaceni 7 NO Sun May 8 11:32:31 CEST 2022
Oznaceni 8 NO Sun May 8 11:32:32 CEST 2022
Oznaceni 9 NO Sun May 8 11:32:33 CEST 2022
Oznaceni 10 NO Sun May 8 11:32:34 CEST 2022
Oznaceni 11 NO Sun May 8 11:32:35 CEST 2022
Oznaceni 12 NO Sun May 8 11:32:36 CEST 2022
Oznaceni 13 NO Sun May 8 11:32:37 CEST 2022
Oznaceni 14 OK Sun May 8 11:32:38 CEST 2022
Oznaceni 15 OK Sun May 8 11:32:39 CEST 2022
Oznaceni 16 OK Sun May 8 11:32:40 CEST 2022
Oznaceni 17 OK Sun May 8 11:32:41 CEST 2022
Oznaceni 18 OK Sun May 8 11:32:42 CEST 2022
Oznaceni 19 OK Sun May 8 11:32:43 CEST 2022
Oznaceni 20 OK Sun May 8 11:32:44 CEST 2022
Oznaceni 21 OK Sun May 8 11:32:45 CEST 2022
Oznaceni 22 OK Sun May 8 11:32:46 CEST 2022
Oznaceni 23 OK Sun May 8 11:32:47 CEST 2022
Oznaceni 24 OK Sun May 8 11:32:48 CEST 2022
Oznaceni 25 OK Sun May 8 11:32:49 CEST 2022
Oznaceni 26 OK Sun May 8 11:32:50 CEST 2022
Oznaceni 27 OK Sun May 8 11:32:51 CEST 2022
Oznaceni 28 OK Sun May 8 11:32:52 CEST 2022
Oznaceni 29 OK Sun May 8 11:32:54 CEST 2022
Oznaceni 30 OK Sun May 8 11:32:55 CEST 2022
[root@Node01 home]#
```

Obrázek 27 Log skriptu [zdroj: vlastní]

Log `pacemaker` obsahuje spoustu dat a informací, obsahuje hodně diff položek, pomocí kterých zkoumá, jestli už daný agent služby přelil potřebná data a konfiguraci na druhý uzel. [Obrázek z výstupu logu se nachází v příloze – Obrázek 1 – Nefiltrovaný výstup `Pacemakeru`]

Důležité informace z logu:

May 08 11:32:26 Node01 pacemaker-based [1833] (cib_perform_op) info: +
/cib/configuration/nodes/node[@id='2']/instance_attributes[@id='node02
instance_attributes']/nvpair[@id='node02-instance_attributes-standby']: @value=on

V čase 11:32:26 tedy přišel povel k odstavení druhého uzlu. K ostatním informacím bylo využito grepnutí logu na string „INFO“, který poskytnul přehlednější informace. Když daná služba totiž v klastru nastartuje nebo změní stav, zahlásí do logu pod svým PID startovací nebo INFO hlášku.

```
May 08 11:32:31 LVM-activate(my_lvm)[7882]: INFO: Activating bc_vg
May 08 11:32:31 LVM-activate(my_lvm)[7882]: INFO: bc_vg: activated successfully.
May 08 11:32:32 Filesystem(my_fs)[7960]: INFO: Running start for /dev/bc_vg/bc_lv on /var/www
May 08 11:32:32 apache(httpd)[8060]: INFO: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.1.11. Set the 'ServerName' directive globally to suppress this message
May 08 11:32:32 mysql(mysql_mysql)[8340]: INFO: MySQL is not running
May 08 11:32:32 mysql(mysql_mysql)[8340]: INFO: MySQL is not running
May 08 11:32:37 mysql(mysql_mysql)[8340]: INFO: MySQL started
May 08 11:32:37 IPAddr2(vip_web)[8727]: INFO: Adding inet address 192.168.1.14/28 with broadcast address 192.168.1.15 to device enp0s8 (with label enp0s8:1)
May 08 11:32:37 IPAddr2(vip_web)[8727]: INFO: Bringing device enp0s8 up
May 08 11:32:37 IPAddr2(vip_web)[8727]: INFO: /usr/libexec/heartbeat/send_arp -i 200 -r 5 -p /run/resource-agents/send_arp-192.168.1.14 enp0s8 192.168.1.14 auto not_used not_used
May 08 11:32:41 IPAddr2(vip_web)[8727]: INFO: ARPING 192.168.1.14 from 192.168.1.14 enp0s8
[root@Node01 ~]#
```

Obrázek 28 Filtrovaný výstup Pacemakeru [zdroj: vlastní]

V čase 11:32:31 tedy aktivoval LVM bc_vg , v čase 11:32:32 připojil LVM do adresáře /var/www a připojil souborový systém. Apache nastartoval během chvíle také, akorát zahlásil warning zprávu, že v konfiguraci chybí nepovinná direktiva „ServerName“ a začal startovat databázi. Databáze se nastartovala v čase 11:32:37 a ve stejném čase také přidal VIP k uzlu.

Uvedené časy souhlasí s časy, které měřily nedostupnost pomocí skriptu a je možné tedy tvrdit, že nedostupnost trvala zhruba 11 sekund.

Pohled na klastr z Crmsh monitoring vypadá následovně:

```
[root@Node01 ~]# crm_mon -1
Cluster Summary:
* Stack: corosync
* Current DC: node02 (version 2.1.0-8.el8-7c3f660707) - partition with quorum
* Last updated: Sun May 8 13:52:07 2022
* Last change: Sun May 8 13:51:52 2022 by root via crm_attribute on node02
* 2 nodes configured
* 6 resource instances configured

Node List:
* Node node02: standby
* Online: [ node01 ]

Active Resources:
* Resource Group: groupweb:
* my_lvm (ocf::heartbeat:LVM-activate): Started node01
* my_fs (ocf::heartbeat:Filesystem): Started node01
* vip_web_arp (ocf::heartbeat:SendArp): Started node01
* httpd (ocf::heartbeat:apache): Started node01
* mysql_mysql (ocf::heartbeat:mysql): Started node01
* vip_web (ocf::heartbeat:IPAddr2): Started node01
[root@Node01 ~]#
```

Obrázek 29 Crmsh monitoring při odstavení uzlu [zdroj: vlastní]

4.5.2 Výpadek při tvrdém vypnutí uzlu

Tato simulace byla také měřena skriptem job.sh a porovnávána s logem pacemakeru na základě dat z uzlu Node02. Uzel Node01 byl vypnut násilně z prostředí hypervizoru, tak aby nedostal žádný čas na postupné vypínání služeb a přípravu na vypnutí.

Skript zachytil výpadek v čase 13:14:23 a správnou odpověď dostal v čase 13:14:36, tedy přibližně 13 sekund byly služby nedostupné na webserveru.

Oznaceni	1	OK	Sun	May	8	13:14:20	CEST	2022
Oznaceni	2	OK	Sun	May	8	13:14:21	CEST	2022
Oznaceni	3	OK	Sun	May	8	13:14:22	CEST	2022
Oznaceni	4	NO	Sun	May	8	13:14:23	CEST	2022
Oznaceni	5	NO	Sun	May	8	13:14:24	CEST	2022
Oznaceni	6	NO	Sun	May	8	13:14:25	CEST	2022
Oznaceni	7	NO	Sun	May	8	13:14:26	CEST	2022
Oznaceni	8	NO	Sun	May	8	13:14:27	CEST	2022
Oznaceni	9	NO	Sun	May	8	13:14:28	CEST	2022
Oznaceni	10	NO	Sun	May	8	13:14:29	CEST	2022
Oznaceni	11	NO	Sun	May	8	13:14:30	CEST	2022
Oznaceni	12	NO	Sun	May	8	13:14:31	CEST	2022
Oznaceni	13	NO	Sun	May	8	13:14:33	CEST	2022
Oznaceni	14	NO	Sun	May	8	13:14:34	CEST	2022
Oznaceni	15	NO	Sun	May	8	13:14:35	CEST	2022
Oznaceni	16	NO	Sun	May	8	13:14:36	CEST	2022
Oznaceni	17	OK	Sun	May	8	13:14:36	CEST	2022
Oznaceni	18	OK	Sun	May	8	13:14:37	CEST	2022
Oznaceni	19	OK	Sun	May	8	13:14:38	CEST	2022
Oznaceni	20	OK	Sun	May	8	13:14:39	CEST	2022
Oznaceni	21	OK	Sun	May	8	13:14:40	CEST	2022
Oznaceni	22	OK	Sun	May	8	13:14:41	CEST	2022
Oznaceni	23	OK	Sun	May	8	13:14:42	CEST	2022
Oznaceni	24	OK	Sun	May	8	13:14:43	CEST	2022
Oznaceni	25	OK	Sun	May	8	13:14:44	CEST	2022
Oznaceni	26	OK	Sun	May	8	13:14:46	CEST	2022
Oznaceni	27	OK	Sun	May	8	13:14:47	CEST	2022
Oznaceni	28	OK	Sun	May	8	13:14:48	CEST	2022
Oznaceni	29	OK	Sun	May	8	13:14:49	CEST	2022
Oznaceni	30	OK	Sun	May	8	13:14:50	CEST	2022

Obrázek 30 Log skriptu [zdroj: vlastní]

Na druhém uzlu klastru lze pozorovat v čase 13:14:29, že začal považovat Node01 za uzel, který opustil klastr a nyní je offline. Dále přepíná „member source“ a maže uzel Node01 z membership a quorum cache.

May 08 13:14:29 Node02 pacemaker-based	[1835]	(pcmk_cpg_membership)	info: Group cib event 2: node01 (node 1 pid 1833) left via cluster exit
May 08 13:14:29 Node02 pacemaker-based	[1835]	(crm_update_peer_proc)	info: pcmk_cpg_membership: Node node01[1] - corosync-cpg is now offline
May 08 13:14:29 Node02 pacemaker-based	[1835]	(update_peer_state_iter)	notice: Node node01 state is now lost nodeid=1 previous=member source=crm_update_peer_proc
May 08 13:14:29 Node02 pacemaker-based	[1835]	(crm_reap_dead_member)	info: Removing node with name node01 and id 1 from membership cache
May 08 13:14:29 Node02 pacemaker-based	[1835]	(reap_crm_member)	notice: Purged 1 peer with id=1 and/or uname=node01 from the membership cache
May 08 13:14:29 Node02 pacemaker-based	[1835]	(pcmk_cpg_membership)	info: Group cib event 2: node02 (node 2 pid 1835) is member
May 08 13:14:29 Node02 pacemaker-control	[1840]	(quorum_notification_cb)	info: Quorum retained membership=559 members=1
May 08 13:14:29 Node02 pacemaker-control	[1840]	(update_peer_state_iter)	notice: Node node01 state is now lost nodeid=1 previous=member source=pcmk__reap_unseen_no

Obrázek 31 Výstup Pacemakeru – dotazování na offline uzel [zdroj: vlastní]

Z logu Pacemakeru nejde přesně určit, kdy začal výpadek, jelikož tuto informaci poskytuje Corosync, který hlídá a spravuje členství uzlů a kvorum v klastru. Z logu Corosyncu lze dohledat, že již v čase 13:14:24 ztratil spojení s prvním uzlem a v čase 13:14:29 ukončil s prvním uzlem spojení a upravil počet členů v klastru.

```

May 08 13:14:24 [1215] Node02 corosync info [KNET ] link: host: 1 link: 0 is down
May 08 13:14:24 [1215] Node02 corosync info [KNET ] host: host: 1 (passive) best link: 0 (pri: 1)
May 08 13:14:24 [1215] Node02 corosync warning [KNET ] host: host: 1 has no active links
May 08 13:14:25 [1215] Node02 corosync notice [TOTEM ] Token has not been received in 2251 ms
May 08 13:14:25 [1215] Node02 corosync notice [TOTEM ] A processor failed, forming new configuration: token timed out (3000ms), waiting 3600ms for consensus.
May 08 13:14:29 [1215] Node02 corosync notice [QUORUM] Sync members[1]: 2
May 08 13:14:29 [1215] Node02 corosync notice [QUORUM] Sync left[1]: 1
May 08 13:14:29 [1215] Node02 corosync notice [TOTEM ] A new membership (2,22f) was formed. Members left: 1
May 08 13:14:29 [1215] Node02 corosync notice [TOTEM ] Failed to receive the leave message. failed: 1
May 08 13:14:29 [1215] Node02 corosync notice [QUORUM] Members[1]: 2
May 08 13:14:29 [1215] Node02 corosync notice [MAIN ] Completed service synchronization, ready to provide service.

```

Obrázek 32 Corosync log [zdroj: vlastní]

K zapnutí služeb na druhém nodu potom docházelo v podobném pořadí a s podobným časovým rozptylem jako při první simulaci s odstavením.

```

May 08 13:14:30 LVM-activate(my_lvm)[6799]: INFO: Activating bc_vg
May 08 13:14:30 LVM-activate(my_lvm)[6799]: INFO: bc_vg: activated successfully.
May 08 13:14:30 Filesystem(my_fs)[6877]: INFO: Running start for /dev/bc_vg/bc_lv on /var/www
May 08 13:14:31 apache(httpd)[6980]: INFO: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 192.168.1.12. Set the 'ServerName' directive globally to suppress this message
May 08 13:14:31 mysql(mysql_mysql)[7259]: INFO: MySQL is not running
May 08 13:14:31 mysql(mysql_mysql)[7259]: INFO: MySQL is not running
May 08 13:14:35 mysql(mysql_mysql)[7259]: INFO: MySQL started
May 08 13:14:36 IPaddr2(vip_web)[7647]: INFO: Adding inet address 192.168.1.14/28 with broadcast address 192.168.1.15 to device enp0s8 (with label enp0s8:1)
May 08 13:14:36 IPaddr2(vip_web)[7647]: INFO: Bringing device enp0s8 up
May 08 13:14:36 IPaddr2(vip_web)[7647]: INFO: /usr/libexec/heartbeat/send_arp -i 200 -r 5 -p /run/resource-agents/send_arp-192.168.1.14 enp0s8 192.168.1.14 auto not_used not_used
May 08 13:14:40 IPaddr2(vip_web)[7647]: INFO: ARPING 192.168.1.14 from 192.168.1.14 enp0s8
[root@Node02 home]#

```

Obrázek 33 Filtrovaný výstup Pacemakeru [zdroj: vlastní]

Ve srovnání s první simulací tedy k přepnutí a obnovení služeb došlo o dvě až tři sekundy později. Toto zpoždění je možným důsledkem toho, že se čekalo, než Corosync prohlásí uzel za neaktivní.

Pohled na klastř z Crmsh monitoring vypadá následovně:

```

[root@Node02 home]# crm_mon -1
Cluster Summary:
* Stack: corosync
* Current DC: node02 (version 2.1.0-8.el8-7c3f660707) - partition with quorum
* Last updated: Sun May 8 13:15:21 2022
* Last change: Sun May 8 12:31:40 2022 by root via crm_attribute on node02
* 2 nodes configured
* 6 resource instances configured

Node List:
* Online: [ node02 ]
* OFFLINE: [ node01 ]

Active Resources:
* Resource Group: groupweb:
* my_lvm (ocf::heartbeat:LVM-activate): Started node02
* my_fs (ocf::heartbeat:Filesystem): Started node02
* vip_web_arp (ocf::heartbeat:SendArp): Started node02
* httpd (ocf::heartbeat:apache): Started node02
* mysql_mysql (ocf::heartbeat:mysql): Started node02
* vip_web (ocf::heartbeat:IPaddr2): Started node02
[root@Node02 home]#

```

Obrázek 34 Crmsh monitoring při vypnutí uzlu [zdroj: vlastní]

5 ZABEZPEČENÍ

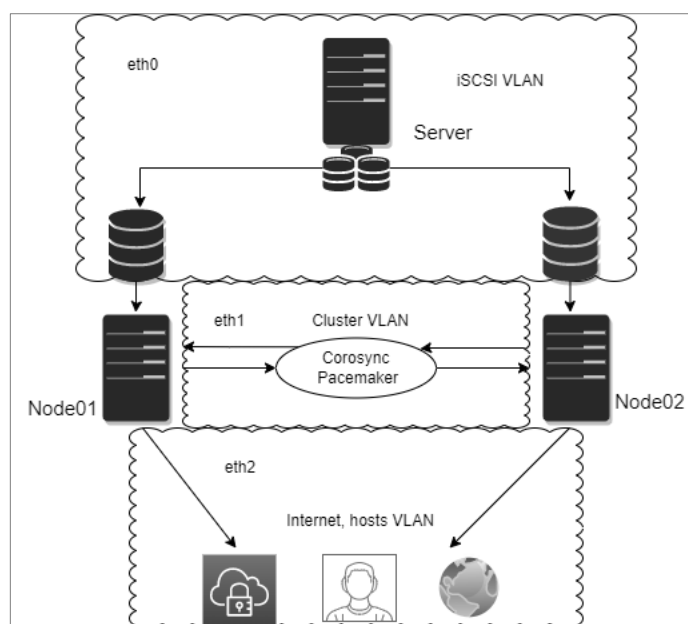
Práci je možno dále rozšířit například zavedením některé z uvedených možností zabezpečení. Zabezpečení bude rozčleněno do tří úrovní, podle kritičnosti.

5.1 Důležité zabezpečení

5.1.1 Dobře nastavená politika hesel

Dobrá politika hesel není důležitá pouze pro uživatele, kteří se přihlašují do svých osobních počítačů nebo aplikačních služeb. U serverů musí správci také zajistit, aby uživatelé používali dostatečně silná hesla. Tato konfigurace činí servery mnohem odolnějšími vůči útokům. Používaná hesla by měla být nad určitou hranicí triviality, například alespoň 12 znaků s náhodnou kombinací písmen, číslic a symbolů. Je rozumné také zvážit implementaci nástroje pro správu hesel, který dokáže ověřit úroveň zabezpečení hesla nebo rovnou vygenerovat dostatečně složité heslo [50]. Ať už je nastaveno jakkoli složité heslo, je důležitá pravidelná aktualizace hesla pro všechny aplikace a přihlašovací údaje, zejména ty s přístupem k administrativnímu serveru. Každé heslo je prolomitelné, u složitějších je to jen otázka času. Většina distribucí Linuxu ve výchozím nastavení obsahuje nástroj pro úpravu informací o vypršení platnosti hesla a stárnutí. Tento program může uživatele donutit k resetování hesla v pravidelných intervalech [51,52].

5.1.2 Logické oddělení sítí



Obrázek 35 Schéma pokročilejšího zapojení [zdroj: vlastní]

Virtuální LAN (Local Area Network) slouží k logickému rozdělení sítě nezávisle na fyzickém uspořádání. Umožňuje tedy síť segmentovat na menší sítě uvnitř fyzické struktury původní sítě. Jinak řečeno, pomocí VLAN je možné dosáhnout stejného efektu, jako když je skupina zařízení připojených do jednoho switchu a druhá skupina zařízení do jiného switchu. Vzniknou dvě nezávislé sítě, které spolu nemohou komunikovat. Pomocí VLAN je možné takovéto dvě sítě vytvořit na jednom switchi. Tohle řešení se používá běžně například v restauracích, kavárnách a dalších veřejně přístupných zařízeních, kdy jedna síť slouží pro potřeby zákazníku a druhá je určena pro vnitropodnikové použití [53,54].

Praktické výhody VLAN

- Snížení broadcastů – hlavní výhodou VLAN je vytvoření více, ale menších broadcastových domén, díky kterým dojde ke zlepšení výkonu sítě snížením provozu.
- Zjednodušená správa – k přesunu zařízení do jiné sítě stačí překonfigurovat zařazení do VLANy, tedy správce konfiguruje SW (zařazení do VLAN) a ne HW (fyzické přepojení).
- Oddělení provozu – jak je vidět na Obrázku 27, pomocí VLAN je možné oddělit síť a tím zvýšit zabezpečení a zároveň to může pomoci vyhnout se problému, který může být způsoben například nějakým útokem, který přijde z internetu a bude se snažit zatížit síť. Když by iSCSI, klastr i aplikace měli svou vlastní síť, nehrozilo by, že při přehlcení linky do internetu dojde k výpadkům spojení například mezi uzly klastru a tím k problémům s komunikací mezi nimi.

5.1.3 Vygenerování SSH klíčů

V této práci je metoda přihlašování pomocí páru klíčů SSH implementována, ale jistě stojí za připomenutí v této kapitole. Páry klíčů SSH, i když nemusí působit uživatelsky přívětivě jako hesla, jsou výrazně bezpečnější. Toto zvýšené zabezpečení lze přičíst šifrování používanému jak přihlášeným serverem, tak i používaným počítačem. Pár klíčů SSH představuje minimálně ekvivalent 12místného hesla, ve skutečnosti je však naprostá většina párů klíčů SSH ještě bezpečnější [51].

5.2 Vyšší stupeň zabezpečení

Důležité zabezpečení je možno ještě rozšířit o další vlastnosti, které přidá tato kapitola.

5.2.1 Zakázat přihlášení pod uživatelem root

Distribuce Linuxu zahrnují superuživatele zvaného „root“, který obsahuje zvýšená oprávnění správce. Ponechání zapnutého přihlášení root může představovat bezpečnostní riziko na serveru, protože hackeři mohou toto pověření zneužít k přístupu na server. K povýšení zabezpečení serveru je možné toto přihlášení zakázat, protože tento uživatel je často terčem útoků [52].

5.2.2 Aktualizace

Bezpečný systém by neměl obsahovat neopravené balíčky, protože ty představují kritické zranitelnosti systému, které by mohli zneužít hackeři. Mnoho distribucí je aktualizováno v průběžném distribučním cyklu s dlouhodobými i krátkodobými verzemi. Systémový administrátor by měl od začátků zvážit, jak provádět aktualizace a jaký provozovat software a podle toho nakonfigurovat příslušné zásady aktualizací. Je možné také stahovat aktualizace pravidelně, ale na stroje je umisťovat a aktualizovat například jednou měsíčně, pokud se neobjeví kritická zranitelnost. Je žádoucí si vždy nechat předchozí i stávající verzi balíčků odložené na nějakém zálohovaném stroji, kdyby náhodou byl aktualizovaný balík vadný [52].

5.2.3 Firewall

Na každém bezpečném serveru by měl být spuštěn firewall jako počáteční obranná linie proti neoprávněným nebo škodlivým žádostem o připojení. Umožňuje uživatelům řídit příchozí síťový provoz na hostitelských počítačích definováním sady pravidel brány firewall. Tato pravidla se používají k třídění příchozího provozu a jeho blokování nebo povolení.

Firewalld je démon služby firewall. Jelikož je dynamický, umožňuje vytvářet, měnit a mazat pravidla bez nutnosti restartovat démona při každé změně pravidel. Firewalld využívá koncepty zón a služeb, které zjednodušují řízení provozu. Povolený provoz závisí na síti, ke které je počítač připojen a na úrovni zabezpečení, která je této síti přiřazena. Služby firewallu jsou předdefinovaná pravidla, která pokrývají všechna potřebná nastavení umožňující příchozí provoz pro konkrétní službu a platí v rámci zóny. Služby používají jeden nebo více portů nebo adres pro síťovou komunikaci. Firewalldy filtrují komunikaci na základě portů.

Jestliže jsou porty pro službu otevřené, pak je síťový provoz povolen. Firewalld blokuje veškerý provoz na portech, které nejsou explicitně nastaveny jako otevřené, ovšem některé zóny, například důvěryhodné, ve výchozím nastavení povolují veškerý provoz [55].

5.3 Nejvyšší stupeň zabezpečení

5.3.1 Nftables

V Red Hat Enterprise Linux 8 je preferovaným řešením firewallu backend úrovně nftables. V RHEL 8 nahrazuje nftables iptables jako výchozí nástroj pro filtrování síťových paketů Linuxu. Nftables je také určeným nástupcem nástrojů ip6tables, arptables, ebtables a ipset. Nabízí četná vylepšení, co se týče většího pohodlí, funkcí a výkonu oproti předchozím nástrojům pro filtrování paketů, zejména [56]:

- Vestavěné vyhledávací tabulky namísto lineárního zpracování
- Jednotný rámec pro protokoly IPv4 a IPv6
- Všechna pravidla se aplikují atomicky namísto načítání, aktualizace a ukládání kompletní sady pravidel
- Podpora pro ladění a trasování v sadě pravidel (nfttrace) a sledování událostí trasování (v nástroji nft)
- Konzistentnější a kompaktnější syntaxe, žádná rozšíření specifická pro protokol
- Netlink API pro aplikace třetích stran

Firewalld funguje jako frontend pro nftables. Firewalld se používá pro jednoduché případy použití brány firewall. Tento nástroj se snadno používá a pokrývá typické případy použití pro tyto scénáře. Nftables slouží k nastavení složitých a výkonově kritických firewallů, například pro celou síť [56].

5.3.2 SELinux

Security Enhanced Linux (SELinux) implementuje řízení povinného přístupu (Mandatory Access Control). Každý proces a systémový prostředek má speciální bezpečnostní štítek nazývaný kontext SELinux. Kontext SELinux, někdy označovaný jako SELinux label, je identifikátor, který abstrahuje detaily na systémové úrovni a zaměřuje se na bezpečnostní vlastnosti entity. Nejenže to poskytuje konzistentní způsob odkazování na objekty v zásadě SELinux, ale také odstraňuje jakoukoli nejednoznačnost, kterou lze nalézt v jiných metodách identifikace. Soubor může mít například více platných názvů cest v systému, který využívá

připojení k vazbě. SELinux zásady používají tyto kontexty v řadě pravidel, která definují, jak mohou procesy interagovat mezi sebou navzájem a s různými systémovými prostředky. Ve výchozím nastavení zásada nepovoluje žádnou interakci, pokud pravidlo výslovně neudělí přístup. Pravidla zásad SELinux definují také to, jak se procesy vzájemně ovlivňují. Přístup je povolen pouze v případě, že existuje pravidlo zásad SELinux, které jej konkrétně umožňuje [57].

Pokud je proces kompromitován, útočník má přístup pouze k normálním funkcím tohoto procesu a k souborům, ke kterým byl proces nakonfigurován. Pokud je například ohrožen Apache HTTP Server, útočník nemůže tento proces využít ke čtení souborů v domovských adresářích uživatele, pokud nebylo přidáno nebo nakonfigurováno konkrétní pravidlo zásad SELinux, které takový přístup umožňuje [57].

ZÁVĚR

Cílem této práce bylo provést čtenáře problematikou vysoce dostupných klastrů na platformě Linux. V úvodu teoretické části jsou popsány informace z oblasti použití počítačových klastrů, typy a význam jejich použití. Druhá kapitola pojednává o failover klastru, jeho klíčových komponentách a konkrétně o architektuře složené z nástrojů Pacemaker a Corosync.

Třetí kapitola obsahuje popis tří hlavních linuxových distribucí, jejich historii a srovnání rozdílů mezi nimi. V posledních částech třetí kapitoly je popsána změna ve vývoji nových Red Hat distribucí a typy virtualizace.

Ve čtvrté kapitole se čtenář dozví, jak postupovat při návrhu, instalaci a konfiguraci jednotlivých uzlů klastru a serveru, který poskytuje klastru sdílené úložiště. Kapitola 4.2 popisuje pokus o nasazení DRBD, jakožto nástroji pro synchronizaci dat mezi blokovými zařízeními a důvod, proč nebylo DRBD využito. Závěrečné části čtvrté kapitoly se věnují experimentálním testům, které mají za cíl ověřit navržený failover klastr a otestovat dostupnost služeb, které klastr spravuje. Jako služba byl spuštěn webserver napojený na databázi, jako nejčastější příklad implementace na failover klastru.

Poslední kapitola se zabývá možným rozšířením této práce například zabezpečením celého řešení.

SEZNAM POUŽITÉ LITERATURY

- [1] HIGH AVAILABILITY CLUSTER: Technology White Paper. EuroNAS [online]. 2018 [cit. 2021-11-13]. Dostupné z: https://euronas.com/wp-content/uploads/2018/08/euroNAS_HA_Cluster_White_Paper.pdf
- [2] KRÍŽ, Filip. Jak vybrat síťové úložiště NAS [online]. 2022 [cit. 2022-03-05]. Dostupné z: <https://www.arecenze.cz/sitova-uloziste-nas/#jak-vybrat-sitove-uloziste-nas>
- [3] Parallel Sysplex principles [online]. 2022 [cit. 2022-03-05]. Dostupné z: https://www.ibm.com/docs/en/cics-ts/6.1_beta?topic=sysplex-parallel-principles
- [4] SHIN YEO, Chee. Cluster Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers [online]. 24 [cit. 2022-01-05]. Dostupné z: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.125.6274&rep=rep1&type=pdf>
- [5] Moderní clusterové systémy a jejich využití. Prostředky implementace clusterů s vysokou dostupností.: Klastrové projekty v Rusku [online]. 2021 [cit. 2022-03-05]. Dostupné z: <https://wiid.ru/cs/power-supply/sovremennye-klasternye-sistemy-i-ih-ispolzovanie-sredstva-realizacii-high-availability>
- [6] GARG, Shubhika. An Overview of Cluster Computing [online]. In: . 2021 [cit. 2022-03-05]. Dostupné z: <https://www.geeksforgeeks.org/an-overview-of-cluster-computing/>
- [7] RAMASUBRAMANIAN IYER, Arthi. Správa clusterů s vysokou dostupností a vyrovnavání zatížení pro místní bránu dat [online]. 2022 [cit. 2022-03-05]. Dostupné z: <https://docs.microsoft.com/cs-cz/data-integration/gateway/service-gateway-high-availability-clusters>
- [8] Klastrované počítače jsou efektivní klastrová řešení. Rozdělení na systémy vysoké dostupnosti a vysokého výkonu [online]. 2016 [cit. 2022-03-05]. Dostupné z: <https://school38vrn.ru/cs/klasternye-kompyutery-predstavlyayut-soboi-effektivnye-klasternye-resheniya.html>
- [9] Vysoká dostupnost pro databáze [online]. 2021 [cit. 2022-03-05]. Dostupné z: <https://www.ibm.com/docs/cs/control-desk/7.6.1?topic=configuration-high-availability-databases>

- [10] POELKER, Christopher. Storage Area Networks For Dummies. 2nd ed. John Wiley & Sons, 2008, s. 7-21. ISBN 0470385138.
- [11] CHAPTER 1. HIGH AVAILABILITY ADD-ON OVERVIEW: RED HAT TRAINING course [online]. [cit. 2022-03-05]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/high_availability_add-on_overview/ch-introduction-haao
- [12] APPENDIX D. LVM VOLUME GROUP METADATA: RED HAT TRAINING course [online]. [cit. 2022-03-05]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/logical_volume_manager_administration/lvm-metadata
- [13] How to Choose Your Red Hat Enterprise Linux File System [online]. 2020 [cit. 2022-03-05]. Dostupné z: <https://access.redhat.com/articles/3129891>
- [14] CHAPTER 42. GETTING STARTED WITH AN EXT4 FILE SYSTEM: RED HAT TRAINING course [online]. [cit. 2022-03-05]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_file_systems/getting-started-with-an-ext4-file-system_managing-file-systems
- [15] What are the file and file system size limitations for Red Hat Enterprise Linux? [online]. 2020 [cit. 2022-03-05]. Dostupné z: <https://access.redhat.com/solutions/1532>
- [16] NEWELL, Glen. An introduction to Linux Access Control Lists (ACLs): Linux Access Control Lists, or ACLs, can take some getting used to, but they're invaluable for getting a finer-grained control of your Linux filesystem permissions. [online]. 2020 [cit. 2022-03-05]. Dostupné z: <https://www.redhat.com/sysadmin/linux-access-control-lists>
- [17] PETROVIČ, Samuel. The Effects of Age on File System Performance. Brno: Masaryk University, 2017, 89 s. Dostupné také z: <https://is.muni.cz/th/n8kgf/thesis.pdf>. Faculty of Informatics. Advisor Rambousek, Adam.

- [18] sengi. What do "extents" feature do in ext4 filesystem in linux? [online]. 2019 [cit. 2022-03-05]. Dostupné z: <https://unix.stackexchange.com/a/360398>
- [19] EXPORTING NFS SHARES: RHEL 8 Training [online]. 2021 [cit. 2022-03-05]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_file_systems/exporting-nfs-shares_managing-file-systems
- [20] KRANZ, Garry. Serial-attached SCSI (SAS) [online]. 2015 [cit. 2022-03-05]. Dostupné z: <https://www.techtarget.com/searchstorage/definition/serial-attached-SCSI>
- [21] Serial Attached SCSI [online]. 2018 [cit. 2022-03-05]. Dostupné z: https://commons.wikimedia.org/wiki/Category:Serial_Attached_SCSI
- [22] GILBERT, Douglas. The Linux 2.4 SCSI subsystem HOWTO. The Linux Documentation Project [online]. 2004 [cit. 2022-03-05]. Dostupné z: <https://tldp.org/HOWTO/SCSI-2.4-HOWTO/index.html>
- [23] Clusters From Scratch demonstrates Pacemaker 2.1 with Corosync 3. ClusterLabs Pacemaker Documentation [online]. 2021 [cit. 2022-03-05]. Dostupné z: <https://clusterlabs.org/pacemaker/doc/>
- [24] CONFIGURING AND MANAGING HIGH AVAILABILITY CLUSTERS: RED HAT ENTERPRISE LINUX 9.0 BETA [online]. [cit. 2022-03-05]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9-beta/html/configuring_and_managing_high_availability_clusters/index
- [25] Basic systemd concepts [online]. 2022 [cit. 2022-03-05]. Dostupné z: <https://documentation.suse.com/smart/linux/html/concept-systemd/index.html>
- [26] Linux Distribution [online]. [cit. 2022-04-10]. Dostupné z: <https://www.suse.com/suse-defines/definition/linux-distribution/>
- [27] GNU/LINUX DISTRIBUTION TIMELINE 12.2 [online]. 2012 [cit. 2022-04-10]. Dostupné z: <http://futurist.se/gldt/2012/02/20/gnulinix-distribution-timeline-12-2/>
- [28] SMART, Christopher. History of 3 Linux Distributions: Slackware, Debian & Red Hat [online]. 2019 [cit. 2022-04-10]. Dostupné z: <https://www.linuxtoday.com/blog/linux-distributions-history/>
- [29] INGALLS, Sam. Best Linux Distros in 2022: OS [online]. 2022 [cit. 2022-03-05]. Dostupné z: <https://www.serverwatch.com/os/linux-distros/>

- [30] KENNEDY, Patrick. Red Hat Goes Full IBM and Says Farewell to CentOS [online]. 2020 [cit. 2022-03-05]. Dostupné z: <https://www.servethehome.com/red-hat-goes-full-ibm-and-says-farewell-to-centos/>
- [31] KRČMÁŘ, Petr. CentOS je mrtev, ať žije CentOS Stream: distribuce předbíhající ve vývoji RHEL [online]. 2020 [cit. 2022-04-10]. Dostupné z: <https://www.root.cz/clanky/centos-je-mrtev-at-zije-centos-stream-distribuce-predbihajici-ve-vyvoji-rhel/>
- [32] BROWN, Korbin. AlmaLinux vs Rocky Linux [online]. 2021 [cit. 2022-03-05]. Dostupné z: <https://linuxconfig.org/almalinux-vs-rocky-linux>
- [33] CIBULKA, Milan. Automatizovaný deployment linuxových serverů a aplikací na platformě Hyper-V. Zlín: Univerzita Tomáše Bati ve Zlíně, 2021, 71 s. Dostupné také z: <http://hdl.handle.net/10563/46105>. Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky, Ústav informatiky a umělé inteligence. Vedoucí práce Sysel, Martin.
- [34] CONFIGURING AN ISCSI TARGET: Red Hat training course [online]. [cit. 2022-04-19]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_storage_devices/configuring-an-iscsi-target_managing-storage-devices
- [35] CONFIGURING AN ISCSI INITIATOR: Red Hat training course [online]. [cit. 2022-04-19]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/managing_storage_devices/configuring-an-iscsi-initiator_managing-storage-devices
- [36] RED HAT HIGH AVAILABILITY ADD-ON CONFIGURATION AND MANAGEMENT OVERVIEW: Red Hat training course [online]. [cit. 2022-04-19]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/cluster_administration/ch-overview-ca
- [37] DRBD: Documentation SUSE [online]. [cit. 2022-04-19]. Dostupné z: <https://documentation.suse.com/sle-ha/15-SP1/html/SLE-HA-all/cha-ha-drbd.html>
- [38] BÖHMWALDER, Christoph. Common administrative tasks: linbit-documentation [online]. 2021 [cit. 2022-04-19]. Dostupné z: <https://github.com/LINBIT/linbit-documentation/blob/master/UG8.4/en/administration.adoc>

- [39] The DRBD9 User's Guide [online]. 2021 [cit. 2022-04-19]. Dostupné z: https://linbit.com/drbd-user-guide/drbd-guide-9_0-en/
- [40] PRABHAKARAN, Pratheek. How to set up a Pacemaker cluster for high availability Linux: Learn how to leverage Red Hat Enterprise Linux Pacemaker for High Availability [online]. 2021 [cit. 2022-04-19]. Dostupné z: <https://www.redhat.com/sysadmin/rhel-pacemaker-cluster>
- [41] GitHub - ClusterLabs/crmsh: command-line interface for High-Availability cluster management on GNU/Linux systems [online]. 2022 [cit. 2022-04-28]. Dostupné z: <https://github.com/ClusterLabs/crmsh>
- [42] ROTH, Tanja a Thomas SCHRAITL. Configuring and Managing Cluster Resources: Command Line [online]. 2022 [cit. 2022-05-06]. Dostupné z: <https://documentation.suse.com/sle-ha/12-SP4/html/SLE-HA-all/cha-ha-manual-config.html>
- [43] BOSKO, Marijan. Server Operating Systems: Server OS Types & How to Choose [online]. 2022 [cit. 2022-04-28]. Dostupné z: <https://phoenixnap.com/kb/server-operating-system>
- [44] BOYETT, Richard. What Is Apache? An In-Depth Overview of Apache Web Server [online]. 2022 [cit. 2022-04-28]. Dostupné z: <https://www.hostinger.com/tutorials/what-is-apache>
- [45] JANOVSKEÝ, Dušan. PHP -- Jak začít [online]. [cit. 2022-04-28]. Dostupné z: <https://www.jakpsatweb.cz/php/jak-zacit.html>
- [46] VOJÁČEK, Zdeněk. SERIÁL „ZAČÍNÁME S PROGRAMOVÁNÍM“: Úvod: Z čeho se skládá web [online]. 2014 [cit. 2022-04-28]. Dostupné z: <https://www.jakpsatweb.cz/php/jak-zacit.html>
- [47] ŠTRAUCH, ADAM. Databáze MariaDB válcuje MySQL [online]. 2012 [cit. 2022-04-28]. Dostupné z: <https://www.root.cz/clanky/databaze-mariadb-valcuje-mysql/>
- [48] Wikipedie: MariaDB [online]. [cit. 2022-04-28]. Dostupné z: <https://cs.wikipedia.org/wiki/MariaDB>
- [49] People Behind MariaDB [online]. 2011 [cit. 2022-04-28]. Dostupné z: <https://mariadb.com/kb/en/people-behind-mariadb/>
- [50] EMPEY, CHARLOTTE. Blog Avast: Jak si nastavit silné heslo [online]. 2019 [cit. 2022-05-05]. Dostupné z: <https://blog.avast.com/cs/jak-si-nastavit-silne-heslo>

- [51] DAVYDOV, Yevgeniya. Linux Server Security: 10 Linux Hardening & Security Best Practices [online]. 2020 [cit. 2022-05-05]. Dostupné z: <https://securityboulevard.com/2020/08/linux-server-security-10-linux-hardening-security-best-practices/>
- [52] AVAST BUSINESS TEAM: How to secure your Linux server [online]. 2021 [cit. 2022-05-05]. Dostupné z: <https://blog.avast.com/secure-your-linux-server-avast>
- [53] Jak vytvořit 2 oddělené sítě [online]. [cit. 2022-05-06]. Dostupné z: <https://www.airwaynet.cz/jak-vytvorit-2-oddelene-site/>
- [54] BOUŠKA, Petr. VLAN - Virtual Local Area Network [online]. 2007 [cit. 2022-05-06]. Dostupné z: <https://www.samuraj-cz.com/clanek/vlan-virtual-local-area-network/>
- [55] Configuring and managing networking: USING AND CONFIGURING FIREWALLD [online]. [cit. 2022-05-06]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/index
- [56] Red Hat training course: GETTING STARTED WITH NFTABLES [online]. [cit. 2022-05-06]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/getting-started-with-nftables_configuring-and-managing-networking
- [57] USING SELINUX: Basic and advanced configuration of Security-Enhanced Linux (SELinux) [online]. [cit. 2022-05-06]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/using_selinux/index

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ARP	Adress Resolution Protocol
ASCII	American Standard Code for Information Interchange
CPU	Central Processing Unit
CSS	Cascading Style Sheets
GNOME	GNU Network Object Model Environment
GNU	GNU's Not Unix
HA	High Availability
HTML	Hypertext Markup Language
IBM	International Business Machines
IT	Information Technology
KDE	K Desktop Environment
KVM	Kernel-based Virtual Machine
LV	Logical Volume
LVM	Logical Volume Management
MySQL	My Structured Query Language
NAS	Network Attached Storage
OS	Operating System
PHP	Hypertext Preprocessor
RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
RPM	Red Hat Package Manager
SAN	Storage Area Network
SAS	Serial Attached Small Computer System Interface
SATA	Serial Advanced Technology Attachment
SSH	Secure Shell

TiB	Tebibyte
VIP	Virtual Internet Protocol
VLAN	Virtual Local Area Network

SEZNAM OBRÁZKŮ

Obrázek 1 Geograficky distribuovaný klastr [zdroj: vlastní]	17
Obrázek 2 Virtualizace - typ 1 [zdroj: vlastní]	40
Obrázek 3 Virtualizace - typ 2 [zdroj: vlastní]	41
Obrázek 4 Schéma virtuálních strojů [zdroj: vlastní]	42
Obrázek 5 iSCSI targetcli [zdroj: vlastní]	44
Obrázek 6 Inicializace iSCSI export [zdroj: vlastní]	44
Obrázek 7 Přidání LUN k exportu [zdroj: vlastní]	45
Obrázek 8 Hotová iSCSI server konfigurace [zdroj: vlastní]	46
Obrázek 9 Příkaz k zobrazení InitiatorName [zdroj: vlastní]	46
Obrázek 10 Ověření připojení LUNy na uzel – 1. možnost [zdroj: vlastní]	47
Obrázek 11 Ověření připojení LUNy na uzel – 2. možnost [zdroj: vlastní]	47
Obrázek 12 Pozice DRBD v rámci linuxového I/O zásobníku [39, https://linbit.com/wp-content/uploads/2020/03/DRBD-Diagram.jpg]	49
Obrázek 13 Výpis Resource Agents [zdroj: 41,42]	52
Obrázek 14 výstup příkazu fdisk – list partition table [zdroj: fdisk]	53
Obrázek 15 Ověření připojení oddílu, zdroj: vlastní	53
Obrázek 16 Ústřížek výpisu crm ra info LVM-activate [zdroj: 41,42]	54
Obrázek 17 Prostředek klastru – LVM [zdroj: vlastní]	54
Obrázek 18 Ústřížek výpisu crm ra info Filesystem [zdroj: 41,42]	54
Obrázek 19 Prostředek klastru – Filesystem [zdroj: vlastní]	54
Obrázek 20 Home page webu [zdroj: vlastní]	57
Obrázek 21 Symbolický odkaz adresářů [zdroj: vlastní]	58
Obrázek 22 Prostředek klastru – VIP a SendARP [zdroj: vlastní]	58
Obrázek 23 Prostředek klastru – Apache [zdroj: vlastní]	58
Obrázek 24 Prostředek klastru – Mysql [zdroj: vlastní]	59
Obrázek 25 Prostředky klastru [zdroj: vlastní]	59
Obrázek 26 Management okno klastru ve webovém prohlížeči [zdroj: vlastní]	60
Obrázek 27 Log skriptu [zdroj: vlastní]	61
Obrázek 28 Filtrovaný výstup Pacemakeru [zdroj: vlastní]	62
Obrázek 29 Crmsh monitoring při odstavení uzlu [zdroj: vlastní]	62
Obrázek 30 Log skriptu [zdroj: vlastní]	63
Obrázek 31 Výstup Pacemakeru – dotazování na offline uzel [zdroj: vlastní]	63

Obrázek 32 Corosync log [zdroj: vlastní]	64
Obrázek 33 Filtrovaný výstup Pacemakeru [zdroj: vlastní]	64
Obrázek 34 Crmsh monitoring při vypnutí uzlu [zdroj: vlastní]	64
Obrázek 35 Schéma pokročilejšího zapojení [zdroj: vlastní]	65

Ve schématech obrázků byly použity ikony od Cisco. [Zdroj:

<https://www.cisco.com/c/en/us/about/brand-center/network-topology-icons.html>]

Obrázky na webserveru byly stáhnuty z Pixabay [Zdroj:

<https://pixabay.com/photos/sunrise-boat-rowing-boat-nobody-1014712/>

<https://pixabay.com/photos/polynesia-french-polynesia-tahiti-3021072/>

<https://pixabay.com/photos/palace-justice-brussels-court-4781577/>

<https://pixabay.com/photos/orion-nebula-emission-nebula-11107/>

<https://pixabay.com/photos/maldives-palm-tree-hammock-beach-3220702/>

<https://pixabay.com/photos/house-pool-interior-design-1477041/>

<https://pixabay.com/photos/monastery-vault-cloister-columns-6560623/>

<https://pixabay.com/photos/architecture-skyscraper-2256489/>

<https://pixabay.com/photos/spring-bird-bird-tit-spring-blue-2295434/>

<https://pixabay.com/photos/trees-wilderness-nature-woods-3822149/>

<https://pixabay.com/photos/butterflies-flowers-pollinate-1127666/>

<https://pixabay.com/photos/beach-birds-sea-ocean-flying-birds-1852945/>]

SEZNAM TABULEK

Tabulka 1. Souborový systém XFS – certifikovaná velikost Red Hat [15].....	21
Tabulka 2. Souborový systém Ext3 – certifikovaná velikost Red Hat [15]	23
Tabulka 3. Souborový systém Ext4 – certifikovaná velikost Red Hat [15]	23
Tabulka 4 Vývojová linie RHEL distribucí [zdroj: vlastní]	37

SEZNAM PŘÍLOH

Obrázek 1 – Nefiltrovaný výstup Pacemakeru

bc_lv_archiv.zip

P I - Konfigurace Corosync

P II – Konfigurace hosts

P III – Skript job.sh

PŘÍLOHA P I: KONFIGURACE COROSYNC

Konfigurační soubor /etc/corosync/corosync.conf

```
totem {
    version: 2
    cluster_name: bc_cluster
    transport: knet
    crypto_cipher: aes256
    crypto_hash: sha256
}

nodelist {
    node {
        ring0_addr: node01
        name: node01
        nodeid: 1
    }

    node {
        ring0_addr: node02
        name: node02
        nodeid: 2
    }
}

quorum {
    provider: corosync_votequorum
    two_node: 1
}

logging {
    to_logfile: yes
    logfile: /var/log/cluster/corosync.log
    to_syslog: yes
    timestamp: on
}
```

PŘÍLOHA P II: KONFIGURACE HOSTS

Konfigurace v /etc/hosts

```
127.0.0.1          localhost    localhost.localdomain  localhost4
localhost4.localdomain4
::1                localhost    localhost.localdomain  localhost6
localhost6.localdomain6
```

```
192.168.1.14 vojtuweb.cz
```

```
192.168.1.10      server
192.168.1.11      Node01
192.168.1.12      Node02
```

PŘÍLOHA P III: SKRIPT JOB.SH

Skript job.sh pro testování

```
#!/bin/bash
OUTPUT='      <h2 class="text-center">Kontakt</h2>'

for i in {1..30};
do echo -n "Oznaceni $i " >> /home/log20.log;
CURL=$(curl --connect-timeout 0.5
http://vojtuweb.cz/hello/testik.php#stavby | grep 'text-
center">Kontakt')

if [[ $CURL = $OUTPUT ]] ; then
    echo -n "OK " >> /home/log20.log
    date >> /home/log20.log; sleep 1;
else
    echo -n "NO " >> /home/log20.log
    date >> /home/log20.log; sleep 0.5;
fi ; done
```