Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Zálohování virtualizační platformy KVM

Plzeň 2021 Patrik Janoušek

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Patrik JANOUŠEK**
Osobní číslo:	**A17B0231P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Zálohování virtualizační platformy KVM**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Nastudujte problematiku virtualizačních platforem a možnosti jejich zálohování a obnovy.
2. Nastudujte dostupné zálohovací systémy pro virtualizační platformu KVM.
3. Navrhněte vhodný systém zálohování pro KVM.
4. Navržené řešení implementujte a ověřte jeho funkčnost.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Luboš Matějka, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **5. října 2020**
Termín odevzdání bakalářské práce: **6. května 2021**

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 26. října 2020

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 11. května 2021

Patrik Janoušek

Abstract

This Bachelor thesis delves into the problem of backup of virtual servers. First, the reader is introduced to a general theory of backup, different types and levels of backups, and problematics of keeping backups consistent. The thesis also explains virtualization, explanation of topics such as hypervisor and its types, virtualization methods, and comparison of available solutions. There is a detailed comparison of a few selected backup solutions at the end of a theoretical part. The practical part of the thesis delves into the design and implementation of a custom backup solution for virtual servers running on QUMU/KVM with a disk in RAW format. And subsequent integration into the Proxmox Virtual Environment. In the end, a custom solution is compared against competing solutions, which offer backups on the Proxmox Virtual Environment.

Abstrakt

Bakalářská práce se zabývá problematikou zálohování virtuálních strojů. Nejprve je čtenář seznámen s obecnou teorií zálohování, jednotlivými typy a úrovněmi záloh, a problematikou zajištění jejich konzistence. Dále se práce zabývá představením virtualizace, vysvětlením pojmu hypervizor a jeho typů, přiblížením virtualizačních metod a porovnání dostupných virtualizačních řešení. V závěru teoretické části je popsáno a srovnáno několik vybraných řešení sloužících pro zálohování virtuálních strojů. Praktická část práce se věnuje návrhu a implementaci vlastního zálohovacího řešení pro virtuální stroje provozované pomocí QEMU/KVM s disky ve formátu RAW, a jeho následné integraci do platformy Proxmox Virtual Environment. V závěru práce je vyvinuté řešení srovnáno s konkurenčními řešeními, které umožňují vytváření záloh virtuálních strojů v Proxmox Virtual Environment.

Obsah

1 Úvod **9**

2 Zálohování **10**
2.1 Typy záloh 10
2.1.1 Plná záloha 10
2.1.2 Diferenciální (rozdílová) záloha 10
2.1.3 Inkrementální (přírůstková) záloha 11
2.2 Sledování změn na disku pomocí dirty bitmapy 12
2.2.1 Dirty bitmapa 12
2.3 Úrovně zálohování 13
2.3.1 Zálohování na úrovni souborů 13
2.3.2 Zálohování na binární úrovni 13
2.3.3 Zálohování na úrovni binárních bloků souborů 14
2.4 Konzistence zálohy 14
2.4.1 Snímek souborového systému 15
2.4.2 Snímek disku 15
2.5 Shrnutí 15

3 Virtualizace **18**
3.1 Model virtualizace 18
3.1.1 Protection ring 18
3.1.2 Domény 19
3.2 Hypervizor 20
3.2.1 Hypervizor prvního typu (nativní) 20
3.2.2 Hypervizor druhého typu (hostovaný) 21
3.3 Typy virtualizace 21
3.3.1 Emulace 22
3.3.2 Paravirtualizace 22
3.3.3 Plná virtualizace 22
3.4 Datová úložiště pro virtuální stroje 23
3.4.1 Uložení dat v souboru 23
3.4.2 LVM 23
3.4.3 LVMTHIN 24
3.4.4 DRBD 24
3.4.5 LINSTOR 24
3.5 Dostupná řešení pro virtualizaci stroje 25

3.5.1 Oracle VM VirtualBox . . . 25
3.5.2 Parallels Desktop . . . 27
3.5.3 QEMU . . . 28
3.5.4 KVM . . . 29
3.5.5 Hyper-V . . . 30
3.5.6 VMware ESXi . . . 33
3.5.7 Proxmox Virtual Environment . . . 34
3.6 Shrnutí . . . 36

4 Zálohování virtuálních strojů 38
4.1 Specifika zálohování virtuálních strojů . . . 38
4.2 Dostupná řešení pro zálohování virtuálních strojů . . . 38
4.2.1 Veeam Backup and Replication . . . 38
4.2.2 Proxmox Backup Server . . . 40
4.2.3 Bacula . . . 41
4.2.4 Acronis Cyber Backup . . . 42
4.3 Shrnutí . . . 43

5 Implementace 46
5.1 Specifikace požadavků . . . 46
5.2 Struktura řešení . . . 46
5.3 Rozšíření funkce dirty bitmapy v QEMU . . . 47
5.3.1 Současná implementace dirty bitmap v QEMU . . . 47
5.3.2 Zpřístupnění dirty bitmap pro externí nástroje . . . 48
5.3.3 Persistence dirty bitmapy pro formát RAW . . . 48
5.4 Nástroj pro provádění záloh . . . 50
5.4.1 Zvolené technologie . . . 50
5.4.2 Struktura aplikace . . . 50
5.4.3 Konfigurace . . . 53
5.4.4 Záloha dat . . . 55
5.4.5 Obnova dat . . . 57
5.5 Integrace do Proxmox Virtual Environment . . . 59

6 Instalace a ověření funkčnosti řešení 62
6.1 Instalace . . . 62
6.1.1 Instalace QEMU v prostředí Proxmox VE . . . 62
6.1.2 Instalace nástroje pro vytváření záloh . . . 63
6.1.3 Integrace řešení do Proxmox VE . . . 64
6.2 Nasazení do prostředí virtuálního stroje . . . 65
6.2.1 Konfigurace virtuálního stroje v Proxmox VE . . . 66

6.2.2 Konfigurace nástroje pro vytváření záloh 67

7 Výkonnostní testy **68**
7.1 Specifikace prostředí 68
7.2 Metodika měření 68
7.3 Měřené scénáře 70
7.3.1 Plná záloha 71
7.3.2 Vytvoření inkrementální/diferenciální zálohy (1 GB změna dat) 73
7.3.3 Vytvoření inkrementální/diferenciální zálohy (3 GB a 30 GB změna dat) 74
7.3.4 Vytvoření inkrementální/diferenciální zálohy (bez změny dat) 76
7.3.5 Vytvoření inkrementální/diferenciální zálohy (změna všech dat) 78
7.3.6 Obnova dat z plné zálohy 79
7.3.7 Obnova dat z inkrementální/diferenciální zálohy - (1 GB změna dat) 81
7.3.8 Obnova dat z inkrementální/diferenciální zálohy - (3 GB a 30 GB změna dat) 82
7.4 Shrnutí 83

8 Závěr **84**

Přehled použitých zkratek

Literatura

A Obsah přiloženého CD

B Návod pro vlastní kompilaci upravené verze QEMU pro Proxmox Virtual Environment

C Uživatelská příručka k nástroji pro vytváření a obnovu záloh
C.1 Vytvoření plné a inkrementální zálohy . . . . .
C.2 Obnovení dat ze zálohy . . . . . .
C.3 Smazání zálohy . . . . . .
C.4 Konfigurace nástroje . . . . . .
C.4.1 Přidání nového virtuálního stroje . . . . . .
C.4.2 Přidání nového virtuálního disku . . . . . .

1 Úvod

S rostoucí digitalizací moderní společnosti se stávají data neustále cennějšími, čímž vznikají stále vyšší požadavky na jejich bezpečné uchování. Z tohoto důvodu je potřeba chránit data před jejich případnou ztrátou či poškozením, a to ať už vlivem hardwaru, softwaru nebo chybou uživatele. Účinnou ochranou proti ztrátě nebo poškození dat je jejich zálohování. Zároveň se také jedná o účinný způsob ochrany dat proti jejich nechtěné změně, jelikož zálohování umožňuje obnovení dat do předchozího stavu.

Dále je v práci popsána problematika virtualizace, která umožňuje rozdělení jednoho fyzického stroje na více virtuálních, díky čemuž dochází k efektivnějšímu využití dostupného hardwaru. Provoz služeb ve virtuálních strojích je v dnešní době rozšířenou problematikou, a tento přístup je čím dál více preferován nad přímým provozem služeb na fyzických strojích. To je podpořeno především skutečností, že je v dnešní době virtualizace na velmi pokročilé úrovni, a má minimální dopad na výkon celkového řešení. Zároveň umožňuje snadnou vertikální škálovatelnost a správu virtuálních strojů, ke které není potřeba fyzický přístup.

Tato práce se zabývá zálohováním virtuálních strojů, a jedná se o kombinaci obou uvedených oborů. Virtuální prostředí přináší v oblasti zálohování nové možnosti, ale i problémy, které je potřeba vyřešit. Jedním z takových problémů je zajištění konzistentního stavu souborového systému v době zálohy.

Cílem této bakalářské práce je navrhnout a implementovat řešení pro plné a inkrementální zálohování virtuálních strojů v prostředí QEMU/KVM [1] využívající virtuálními disky ve formátu RAW [1], které je možné provozovat pomocí správce logických disků LVM [2] nebo distribuovaného síťového úložiště LINSTOR [3]. Poté bude vyvinuté řešení nad rámec zadání integrováno do virtualizační platformy Proxmox Virtual Environment [4] a jejího webového administračního rozhraní. Motivací pro vypracování práce je absence dostupného zálohovacího nástroje s podporou inkrementálního zálohování virtuálních strojů v QEMU/KVM.

2 Zálohování

Zálohováním dat se rozumí proces, kdy jsou data zkopírována na jiné úložiště, díky čemuž je v případě jejich ztráty či poškození možné provést obnovu dat na původní zařízení [5].

2.1 Typy záloh

Zálohy podle rozlišujeme podle typu na plné, inkrementální (přírůstkové) a diferenciální (rozdílové). Výběr vhodného typu zálohy závisí na dostupném diskovém prostoru pro jejich uložení, požadavku na rychlost provedení a důležitosti zálohovaných dat.

2.1.1 Plná záloha

Plná záloha obsahuje kompletní kopii všech dat, a to bez ohledu na to, zda byla data od poslední zálohy změněna či nikoliv.

Výhodou je zejména vzájemná nezávislost záloh, díky které poškození jedné zálohy nemá vliv na ostatní. Další výhodou je snadná obnova dat, jelikož není nutné provádět sloučení více záloh jako je tomu u ostatních popisovaných typů (viz dále).

Největší nevýhodou je velikost jednotlivých záloh, která je shodná jako velikost zálohovaných dat. S tím je spojena i dlouhá doba zálohy a velké vytížení zálohovaného úložiště. [6]

2.1.2 Diferenciální (rozdílová) záloha

Diferenciální záloha je tvořena počáteční plnou zálohou, vůči které jsou v následujících zálohách ukládány pouze změny dat.

Detekci změněných dat je možné dělat dvěma způsoby. První způsob spočívá v provedení dočasné plné zálohy, která je následně porovnána s poslední plnou zálohou. Tímto postupem dojde k získání rozdílu dat, který představuje požadovanou diferenciální zálohu. Nevýhodou této metody je vysoká náročnost na zálohované úložiště, která je stejná jako při použití plné zálohy. Další nevýhodou jsou nároky na úložiště zálohovacího serveru, na které je nutné uložit dočasnou plnou, a následně nově vytvořenou diferenciální zálohu. Tato metoda je zároveň velmi náročná na výpočetní výkon.

Druhým způsobem je aktivní sledování zápisových operací na úložiště a ukládání informací o tom, na která místa proběhl zápis dat. Tímto způsobem je možné v rámci diferenciální zálohy kopírovat pouze data, která byla od poslední zálohy skutečně změněna. Po provedení nové plné zálohy je potřeba informace o změnách dat smazat, aby mohly být ukládány nové informace pro účely další zálohy. Oproti předchozímu způsobu dochází k výrazně menšímu vytížení zálohovaného úložiště, jelikož není potřeba číst i nezměněná data. Dále zálohovací řešení nepotřebuje dostupnou kapacitu úložiště pro uložení dočasné plné zálohy, ale pouze pro nově vzniklou diferenciální zálohu. Nevýhodou je, že v případě ztráty informací o změněných místech na disku je potřeba provést diferenciální zálohu prvním způsobem nebo plnou zálohu.

Obecnou výhodou diferenciálních záloh je ušetření úložného prostoru pro jejich uchování. Nevýhodou je, že pro obnovu dat je potřeba provést sloučení zvolené diferenciální zálohy a předcházející plné zálohy. Dále poškození nebo ztráta plné zálohy vede ke znehodnocení všech následujících diferenciálních záloh až do další plné zálohy. Nevýhodou je také rostoucí velikost diferenciálních záloh se zvyšujícím se počtem změn od poslední plné zálohy. [6]

2.1.3 Inkrementální (přírůstková) záloha

Inkrementální záloha je stejně jako diferenciální tvořena počáteční plnou zálohou, vůči které jsou v následujících zálohách ukládány pouze změny dat od poslední zálohy libovolného typu.

Stejně jako u diferenciální zálohy je možné provádět detekci změn dvěma způsoby. První způsob opět spočívá ve vytvoření dočasné plné zálohy. Tu v tomto případě není možné porovnávat s poslední plnou zálohou, ale je nutné sloučit všechny předchozí inkrementální zálohy až do poslední plné zálohy. Porovnáním sloučené zálohy s dočasnou plnou zálohou získáme rozdíl dat, který představuje novou inkrementální zálohu. Opětovnou nevýhodou je vysoké vytížení zálohovaného úložiště, které je shodné jako u plné zálohy. Další nevýhodou je nutnost vypočtení nové inkrementální zálohy na základě předešlých záloh, což vytěžuje úložiště i procesor.

Druhým způsob je u inkrementální zálohy shodný s tím, který je popsán u diferenciální zálohy. Jediným rozdílem je, že jsou informace o změněných datech mazány po každém vytvoření zálohy libovolného typu. Tím je zajištěno, že každá inkrementální záloha obsahuje data, která byla změněna od poslední zálohy. Při ztrátě informací o změněných místech na disku je potřeba provést inkrementální zálohu první metodou nebo plnou zálohu.

Obecnou výhodou inkrementálního typu záloh je jejích nejmenší velikost. Nevýhodou je, že pokud dojde k poškození jedné zálohy, tak dojde k znehodnocení všech záloh až do následující plné zálohy. Dále poškozením plné zálohy dojde k poškození všech inkrementálních záloh, které z ní vycházejí. Zároveň s rostoucí sekvencí inkrementálních záloh roste výpočetní náročnost obnovy, jelikož je potřeba pro účely obnovy provést sloučení všech záloh až do poslední plné zálohy. [6]

2.2 Sledování změn na disku pomocí dirty bitmapy

Jak již bylo zmíněno v kapitolách 2.1.2 a 2.1.3, tak je možné detekovat změnu dat vůči poslední záloze již v rámci zápisové operace na disk, což je výrazně efektivnější metoda než detekce těchto změn až při požadavku na vytvoření inkrementální či diferenciální zálohy. K uchování informací o změně dat, které jsou zaznamenávány již při jejich zápisu na disk, je využívána datová struktura nazývaná bitmapa, která umožňuje uložení informace pravda či nepravda na úrovni bitů, čímž dochází k velmi efektivnímu využití paměti. Speciálním případem využití této datové struktury je dirty bitmapa.

2.2.1 Dirty bitmapa

Dirty bitmapa je speciální případ využití bitmapy, kdy jsou do této datové struktury ukládány informace o zápisu dat na cílové úložiště. V případě, že dojde k zápisu dat na sledované úložiště, tak je v dirty bitmapě označena hodnotou jedna (pravda) shodná pozice, na kterou ve sledovaném úložišti proběhl zápis. Aby nebylo nutné sledovat změnu každého zapsaného bitu, což by vyžadovalo dirty bitmapu o stejné velikosti jako je velikost úložiště, tak je definován parametr granularita.

Granularita bitmapy

Granularita slouží k určení nejmenší možné sledovatelné jednotky změněných dat. Tento parametr rozdělí sledovaná data na bloky o pevně definované délce, a poté je sledována libovolná změna v celém bloku. Díky tomu dojde ke snížení velikost dirty bitmapy vůči velikosti sledovaného úložiště tolikrát, kolik je hodnota granularity.

Standardně volenou hodnotou granularity je 65536 bajtů, která poskytuje vhodný kompromis mezi velikostí dirty bitmapy a přesností sledování pozic změněných dat.

2.3 Úrovně zálohování

K datům určených k záloze je možné přistupovat na úrovni jednotlivých souborů, binárních dat nebo binárních bloků souborů. Všechny úrovně slouží k vytvoření kopie dat, přičemž každá má vlastní podmínky na zálohovaný stroj a různé možnosti zálohy a obnovy dat.

2.3.1 Zálohování na úrovni souborů

Zálohování na úrovni souborů umožňuje provádět zálohu resp. obnovu konkrétně zvolených souborů či složek na zálohovaném stroji, což vyžaduje, aby zálohovací řešení mělo přístup na zálohovaný stroj a umělo pracovat s používaným souborovým systémem. Tento problém je řešen instalací tzv. agenta, který zajišťuje komunikaci se zálohovacím serverem, a může obsahovat i grafické rozhraní, které umožňuje správu záloh lokálního stroje.

Výhodou je jednoduchost celého řešení, kdy je administrátor schopen snadno procházet obsah jednotlivých záloh, a obnovovat pouze vybrané složky či soubory.

Nevýhodou je rychlost obnovy dat v případě havárie, jelikož se zálohování na této úrovni typicky nevyužívá pro zálohu operačního systému a nainstalovaných aplikací. V této situaci by bylo nutné nejprve nainstalovat operační systém se všemi aplikacemi, a až poté provést obnovu dat. Přesto je možné zálohovat na úrovni souborů kompletní operační systém i s aplikacemi a daty. Výhodou tohoto řešení je možnost obnovení dat do jiného souborového systému s odlišnou konfigurací a velikostí. [7]

2.3.2 Zálohování na binární úrovni

Zálohování na binární úrovni pracuje s celým úložištěm bez znalosti vnitřní struktury dat. V případě, že je záloha úložiště prováděna z vnějšku zálohovaného stroje (např. hostujícím hypervizorem), tak není potřeba instalace agenta přímo na zálohovaný stroj. Pokud je záloha prováděna přímo ze zálohovaného stroje, tak je instalace agenta vyžadována i při využití zálohy na této úrovni. Výsledkem zálohy je soubor s obrazem disku nebo jeho částí v závislosti na typu zálohy (viz 2.1).

Výhodou je rychlá obnova, pokud dojde k havárii celého systému, jelikož plná záloha obsahuje kompletní obraz disku zálohovaného stroje, který je možné připojit k jinému stroji, a obnovit tím provoz služby.

Nevýhodou je absence možnosti jednoduché obnovy konkrétního souboru, pro kterou je potřeba připojit soubor s obrazem disku, což umožní

procházet souborový systém. To je komplikované zejména v případě inkrementálních či diferenciálních záloh, které je potřeba nejprve sestavit do podoby plné zálohy, která obsahuje kompletní obraz disku. [7]

2.3.3 Zálohování na úrovni binárních bloků souborů

Provádění záloh na úrovni bloků souborů kombinuje princip obou zmíněných úrovní. Jednotlivé soubory jsou rozděleny na bloky o předem definované velikosti, a poté je provedena jejich záloha. Kvůli tomu je na rozdíl od zálohování na binární úrovni vyžadována znalost zálohovaného souborového systému, což je řešeno instalací agenta.

Hlavní výhody zálohování na úrovni bloků souborů vynikají především u inkrementálního a diferenciálního typu záloh, jelikož zde není nejmenší jednotkou obsahu zálohy celý soubor, ale pouze jeho blok. Důsledkem je zrychlení procesu zálohy a snížení její velikosti.

Nevýhodou je komplikovaná obnova souboru, jelikož je nutné jeho správné sestavení ze zálohovaných bloků, což má dopad na rychlost obnovy velkých souborů. [7]

2.4 Konzistence zálohy

S rostoucím časem provedení zálohy vzniká riziko změny dat v jejím průběhu, což vede k její nekonzistenci. Typickým projevem je vytvoření nové podoby dat, které spolu v konkrétním čase neexistovaly. K tomuto problému dochází zejména v situaci, kdy jsou zálohována data, do kterých v danou chvíli probíhá zápis. Důsledkem může být poškození souborového systému, uživatelských dat, konfiguračních souborů nebo celého operačního systému.

Ne vždy je potenciální nekonzistence zálohy výrazným nedostatkem, a v některých případech není potřeba tento problém řešit. Pokud dochází k záloze dat, do kterých probíhá zápis zřídka, tak je riziko nekonzistence zálohy velmi nízké, a je možné ho akceptovat. Typickým příkladem jsou data s nízkou frekvencí zápisu, které nejsou často přidávány či editovány. V případě, že dojde k narušení jejich konzistence v jedné záloze, jsou data opravena v následující záloze, pokud do nich v průběhu této zálohy opět neproběhl zápis. Opačnou situací jsou data s vysokou frekvencí zápisu, kde je riziko porušení konzistence velmi vysoké. Pokud dojde k poškození části dat v jedné záloze, tak existuje vysoká šance, že v průběhu následující zálohy dojde k poškození jiné části dat.

2.4.1 Snímek souborového systému

Některé souborové systémy, jako např. btrfs (B-tree file system) [8] nebo ZFS (Zettabyte File System) [9], podporují vytvoření snímku souborového systému. Tímto dojde k okamžitému zmrazení stavu celého souborového systému, na který je i přesto stále možné provádět zápis, jelikož jsou veškeré zápisové operace přesměrovány na odlišnou pozici disku, díky čemuž nedochází k modifikaci původních dat obsažených ve snímku. Toto řešení má výrazný dopad na výkon, jelikož je při čtecí operaci nutné provádět sloučení nově zapsaných dat s těmi, které jsou obsaženy ve snímku.

Výhodou je, že souborový systém má přístup k procesu vytvoření snímku, a může eliminovat šanci na jeho zachycení v nekonzistentním stavu.

Nevýhodou je nutnost používání konkrétního souborového systému, který musí implementovat podporu snímků. Na rozdíl od běžných souborových systémů, jako např. ext4 [10], nemusí tyto souborové systémy splňovat požadavky na výkon nebo stabilitu.

2.4.2 Snímek disku

Existují řešení, která umožňují provádět snímek na úrovni celého disku, čímž zaniká požadavek na konkrétní vlastnosti souborového systému, který v tomto případě není podstatný. Nejběžnějším řešením je LVM (Logical Volume Manager), který nad fyzickými disky vytváří logickou vrstvu, která mimo jiné umožňuje i vytváření snímků.

Zmíněná nezávislost na souborovém systému nemusí být výhodou, jelikož je možné zachytit souborový systém v nekonzistentním stavu, což může vést k potřebě jeho opravy při pokusu o obnovení dat. Tomuto problému lze předejít vynucením synchronizace souborového systému na disk před vytvořením snímku.

2.5 Shrnutí

Nejdůležitějším krokem při výběru zálohovacího řešení je volba správného typu záloh, jelikož tento aspekt ovlivňuje parametry výsledného řešení nejvýraznějším způsobem. Obecně nelze říct, který z popsaných typů je nejlepší, jelikož má každý svá podstatná pozitiva i negativa. Ten, který má nízké nároky na kapacitu úložiště, je vysoce náchylný na poškození většího množství záloh, a naopak.

Plná záloha	Diferenciální záloha	Inkrementální záloha
Data obsažená v záloze		
Všechna zálohovaná data	Obsahuje pouze změněná data od poslední plné zálohy	Obsahuje pouze změněná data od poslední zálohy libovolného typu
Průběh velikosti záloh na základě změny dat		
Stále stejná velikost zálohy	Zvyšující se s objemem změněných dat od poslední plné zálohy	Zvyšující se s objemem změněných dat od poslední zálohy libovolného typu
Rychlost zálohy		
Velmi pomalá (nutné čtení všech dat)	Rychlá (možné číst pouze změněná data od poslední plné zálohy)	Velmi rychlá (možné číst pouze změněná data od poslední zálohy libovolného typu)
Rychlost obnovy		
Velmi rychlá (není vyžadováno sloučení více záloh)	Rychlá (nutnost sloučení poslední plné zálohy s vybranou)	Velmi pomalá (nutnost sloučení všech záloh od poslední plné až do vybrané)
Dopad poškození jedné zálohy		
Při poškození jedné zálohy nedochází k poškození ostatních záloh	Poškození plné zálohy vede k poškození všech závisejících diferenciálních záloh	Poškození libovolné zálohy vede k poškození všech záloh až do následující plné

Tabulka 2.1: Souhrn parametrů popsaných typů záloh

Dalším podstatným parametrem je úroveň, na které jsou zálohy prováděny. Do domácího prostředí je vhodné zálohování na úrovni souborů nebo jejich bloků, jelikož se jedná o uživatelsky velmi jednoduchý způsob, který umožňuje snadno obnovovat či zálohovat jednotlivé soubory. Zálohování na binární úrovni je naopak vhodné pro zálohování serverů, které je nutné v případě havárie co nejrychleji uvést do opětovného provozu. Přesto má zálohování na úrovni souborů využití i v případě serverů, kde se používá především pro zálohu konfiguračních souborů, které je možné v případě potřeby rychle obnovit.

Konzistenci celé zálohy je možné zajistit snímkem souborového systému, kde je souborový systém schopen zabezpečit, že se nebude v době vytváření snímku nacházet v nekonzistentním stavu. Avšak tuto funkcionalitu podporuje jen velmi málo souborových systémů. Druhou variantou je snímek na úrovni disku, který je mnohem univerzálnější, ale není schopen zajistit konzistenci souborového systému, a je nutné tento problém řešit alternativními způsoby. V případě využití virtualizace je možné využít pozastavení stroje, které vynutí synchronizaci dat na disk.

V rámci vyvíjeného řešení (viz 5) budou implementovány následující funkcionality:

- Vytvoření a obnova plných a inkrementálních záloh, která zajistí požadovanou ochranu před ztrátou či poškozením dat při zachování nízké velikosti jednotlivých záloh
- Aktivní sledování změny dat již při jejich zápisu na disk. K uložení informací o změně dat bude využita popisovaná dirty bitmapa. Díky tomu nebude nudné provádět detekci změněných dat od poslední zálohy až při požadavku na její vytvoření, ale bude docházet k přímé záloze pouze změněných dat.
- Podpora pro distribuované síťové úložiště LINSTOR (viz 3.4.5), které na pozadí vytváří snímky disku pomocí již zmíněného LVM, díky čemuž bude zajištěna konzistence záloh. Důvodem zvolení tohoto úložiště je jeho využívání na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

3 Virtualizace

Virtualizace je technologie umožňující vytváření služeb, které jsou běžně spjaty s fyzickým hardwarem. Virtualizovat je možné disky, síťové karty, paměť RAM, ale i celý stroj. Virtualizované prostředky mají své vnější chování shodné s fyzickými, čímž je zajištěna kompatibilita se softwarem využívajícím tyto prostředky. Avšak jejich vnitřní implementace je skryta, a je odlišná od běžných prostředků ve fyzické podobě.

Virtualizace přidává další vrstvu abstrakce nad fyzický stroj. Jejím hlavním využitím je efektivnější využití dostupného hardwaru při zachování co možná největší izolace jednotlivých virtualizovaných prostředí. Zároveň je také možné z administračního rozhraní měnit rozdělení hardwarových prostředků pro jednotlivé virtuální stroje, a není tak pokaždé nutné fyzicky vyměnit nebo doplnit požadovanou komponentu. Aby toto bylo možné, tak je potřeba mít softwarové řešení, které je schopné vytvořit požadovanou vrstvu nad fyzickým hardwarem. Takové softwarové řešení se nazývá hypervizor, a je nejvyšší vrstvou ve virtualizačním modelu.

3.1 Model virtualizace

Ve virtuálním prostředí je potřeba zajistit, aby virtuální stroje nemohly získat kontrolu na hardwarem hostitele. Zároveň virtuální stroj nesmí být schopen ovládat ostatní virtuální stroje, které jsou provozovány na stejném hostiteli. Tato problematika je na procesoru s podporou hardwarově asistované virtualizace řešena pomocí tzv. protection ringu.

3.1.1 Protection ring

Protection ring neboli ochranný kruh definuje čtyři úrovně oprávnění procesoru, které jsou označovány jako ring 0 (nejvíce privilegovaný) až ring 3 (nejméně privilegovaný). Ve více privilegovaném ringu je možné používat více instrukcí procesoru, a zároveň je tento ring chráněn před případnými chybami méně privilegovaného ringu. Ring 0 je určen k běhu operačního systému, ring 1 a 2 je určen pro ovladače hardwarových zařízení, a na ringu 3 jsou spouštěny uživatelské aplikace.

V případě provozu virtuálního prostředí s procesorem bez hardwarové podpory virtualizace běží hypervizor na ringu 0 nebo 1 podle toho, zda se

jedná o hypervizor prvního nebo druhého typu. Pokud hostitelský procesor neimplementuje hardwarovou podporu pro virtualizaci, a virtuální stroj se pokusí spustit instrukci v ringu 0, tak proběhne její emulace, čímž je zabráněno neoprávněnému přístupu na hardware hostitele. Všechny neprivilegované instrukce jsou v obou případech spouštěny přímo na procesoru bez jejich emulace.

Pokud hostitelský procesor podporuje virtualizaci na hardwarové úrovni, tak dojde k vytvoření nového nejvíce privilegovaného ringu -1, který je použit pro provoz hypervizoru. Díky tomu je virtuálnímu stroji s nemodifikovaným operačním systémem umožněno, aby spouštěl privilegované instrukce přímo na ringu 0 bez nutnosti využití emulace nebo paravirtualizace. [11]

3.1.2 Domény

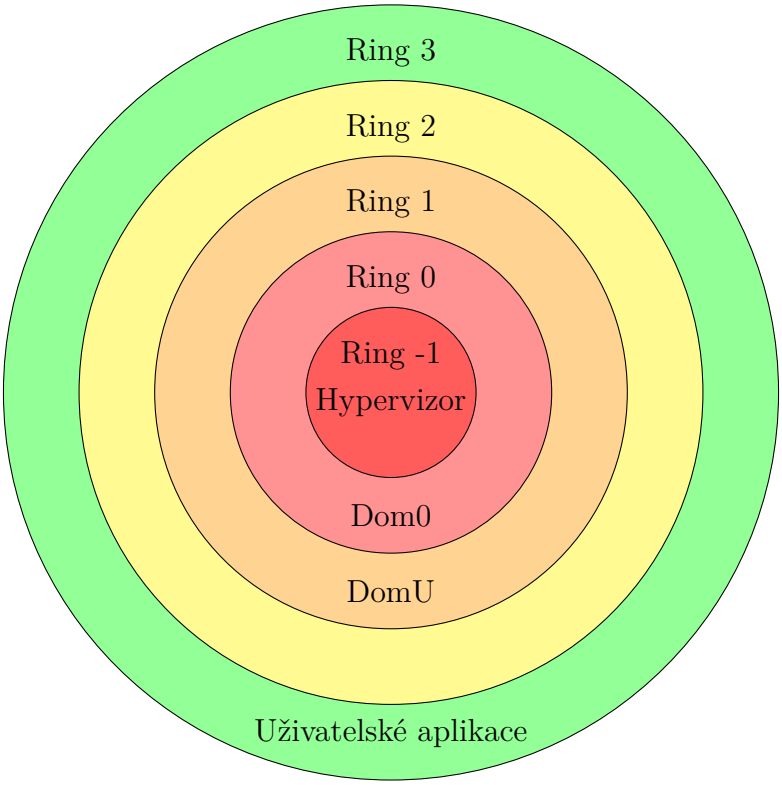
Hypervizory jako Xen nebo Hyper-V (viz 3.5.5) rozdělují virtuální stroje do dvou domén s odlišnými úrovněmi oprávnění vůči hypervizoru.

Hostitelská doména

Hostitelská doména, která je v terminologii hypervizoru Xen označována jako Dom0 nebo Domain 0, je privilegovaný virtuální stroj, který má oprávnění konfigurovat hypervizor a ostatní virtuální stroje (hostované domény, viz dále). Tato doména je na každém hypervizoru spuštěna pouze jednou, přičemž je provozována na ringu 0. [12]

Hostovaná doména

Hostovaná doména, která je v terminologii Xenu označována jako DomU, slouží k provozu neprivilegovaných virtuálních strojů, které nemají vůči hypervizoru žádná oprávnění. Hostované domény jsou provozovány v ringu 1 nebo 3, a může jich být vytvořeno více. [12]



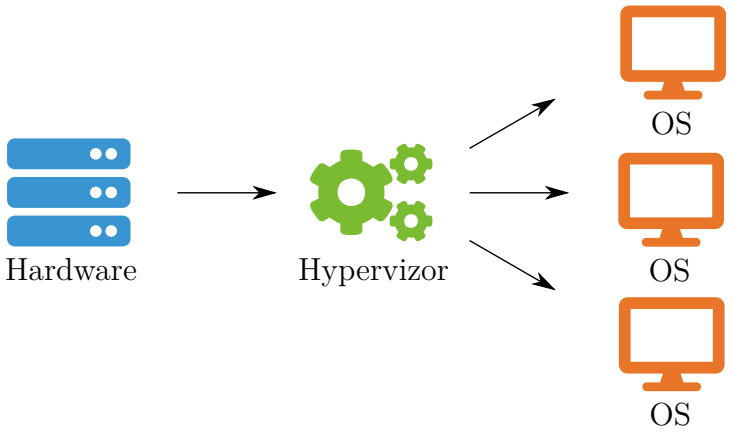
Obrázek 3.1: Protection ring v případě využití virtualizovaného prostředí s hardwarovou podporou virtualizace

3.2 Hypervizor

Hypervizor neboli VMM (Virtual Machine Monitor) je softwarová vrstva zprostředkovávající virtualizaci, která zajišťuje vytvoření a správu virtuálního prostředí využívaného hostovaným virtuálním strojem. Hypervizory dělíme podle běhového prostředí na hypervizor prvního a druhého typu. [13]

3.2.1 Hypervizor prvního typu (nativní)

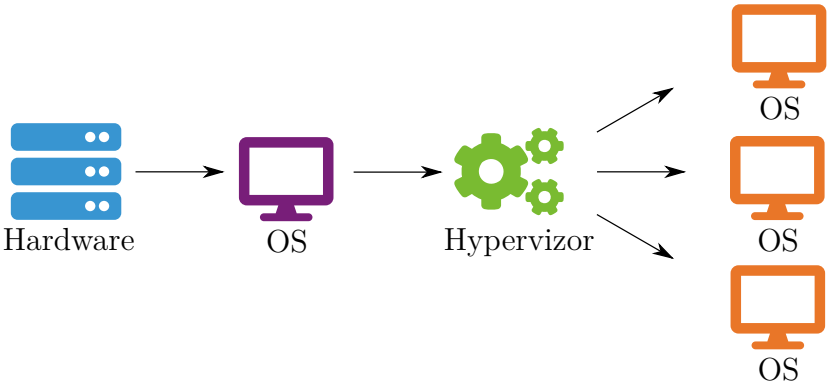
Hypervizor prvního typu běží na nejnižší dostupné úrovni, kterou je hostitelský hardware. Hypervizory prvního typu mají minimální dopad na výkon, a nevyžadují mnoho prostředků pro vlastní běh. Nevýhodou je, že ovladače pro použitý hardware musí být vytvořené přímo pro zvolený hypervizor, což je omezující faktor při jeho výběru.



Obrázek 3.2: Architektura hypervizoru prvního typu

3.2.2 Hypervizor druhého typu (hostovaný)

Hypervizor druhého typu je instalován na operační systém hostitele, a tudíž na rozdíl od hypervizoru prvního typu není potřeba, aby byly ovladače pro hardware napsané přímo pro konkrétní hypervizor. Jelikož je operační systém další vrstvou mezi virtuálním prostředím a hardwarem, tak tento typ hypervizoru neposkytuje stejně vysoký výkon jako hypervizor prvního typu. [13]



Obrázek 3.3: Architektura hypervizoru druhého typu

3.3 Typy virtualizace

Pro vytvoření virtualizovaného prostředí je dostupných několik typů virtualizace. Některé slouží pouze k zprostředkování přístupu ke sdílenému hardwaru, a jiné dokáží simulovat chování aplikace nebo hardwaru, který není

fyzicky dostupný. Základními typy virtualizací jsou emulace, paravirtualizace a plná virtualizace.

3.3.1 Emulace

Za emulaci je označován typ, který umožňuje napodobit chování jiné hardwarové platformy, operačního systému nebo aplikace, díky čemuž je umožněno spuštění nebo sestavení aplikace, která je vytvořena pro jinou hardwarovou platformu. K tomuto účelu je rozšířeno využívání emulátoru QEMU.

Emulace poskytuje oproti ostatním popisovaným typům nejnižší výkon, jelikož je prováděna za běhu emulovaného prostředí.

3.3.2 Paravirtualizace

Paravirtualizace je typ podobný plné virtualizaci, jelikož využívá hypervizor, který zprostředkovává přístup ke sdílenému fyzickému hardwaru. Hostovaný operační systém si je vědom svého běhu ve virtuálním prostředí, a přizpůsobuje tomu své chování tím, že volá nízkoúrovňové funkce hypervizoru, který díky tomu nemusí aktivně odchytávat tyto procesorové instrukce jako v případě plné virtualizace. Nejpoužívanějším hypervizorem podporující paravirtualizaci je Xen.

Výkon paravirtualizace je velmi podobný jako v nevirtualizovaném prostředí, jelikož je zjednodušena virtualizační vrstva, která má největší dopad na snížení výkonu. Nevýhodou tohoto typu virtualizace je nutnost modifikace operačního systému, který podporu pro paravirtualizaci musí explicitně implementovat. [14]

3.3.3 Plná virtualizace

Plná virtualizace, která je také často označována jako nativní virtualizace, poskytuje přístup k hardwarovým prostředkům skrz hypervizor, který v tomto případě běží přímo na úrovni hardware bez účasti operačního systému. Nejpoužívanějšími hypervizory s podporou plné virtualizace jsou VMware ESXi [15], Microsoft Hyper-V [16], KVM [1] a VirtualBox [17].

Největší výhodou tohoto typu virtualizace je absence nutnosti modifikace hostovaného operačního systému, který o běhu ve virtuálním prostředí není informován a nijak se na virtualizaci nepodílí.

Nevýhodou je, že hypervizor musí odchytávat instrukce, které se virtualizovaný operační systém pokusí spustit v privilegovaném režimu procesoru, a zároveň musí poskytnout jejich alternativní implementaci, aby virtuální stroj

skrz tuto instrukci nemohl získat kontrolu nad fyzickým strojem. Zpracování privilegovaných instrukcí má výrazný dopad na výkon, jelikož vyžaduje za běhu překlad, odchycení a emulaci instrukce. Tento výkonnostní dopad snižuje hardwarová podpora virtualizace přímo na straně procesoru, jako např. AMD-V nebo Intel VT-x. [14]

3.4 Datová úložiště pro virtuální stroje

Data virtuálních strojů jsou typicky ve výchozím nastavení hypervizoru ukládána ve formě souborů na hostitelském stroji, které reprezentují jednotlivé virtuální disky. Avšak je tyto data možné ukládat i na pokročilejší úložiště jako jsou LVM nebo LINSTOR, pokud je hypervizorem podporováno.

3.4.1 Uložení dat v souboru

V tomto případě je každý disk virtuálního stroje uložen na souborovém systému hostitelského stroje ve formě jednoho či více souborů. Dostupné funkcionality a technické limitace jsou plně závislé na hostitelském souborovém systému a zvoleném formátu virtuálního disku. Pokud formát disku neumožňuje např. vytváření jeho snímku, tak je komplikované tuto funkcionalitu na úrovni hypervizoru dodat, což by bylo možné při využití souborových systémů btrfs či ZFS. Avšak tyto souborové systémy umožňují pouze vytvoření snímku celého souborového systému, na kterém jsou uloženy virtuální disky všech provozovaných strojů. Vytvoření snímku takto aktivně využívaného úložiště více virtuálními stroji by mohlo vést k výraznému poklesu výkonu celého úložiště, a tudíž bylo by nutné ukládat každý virtuální disk na samostatném souborovém systému, čímž by bylo umožněno vytvoření snímku konkrétního virtuálního disku. Nicméně souborové systémy s podporou pro vytváření snímků nemusí být hypervizorem podporovány, a tak je tato možnost přidání podpory vytváření snímků spíše teoretická, a její reálná implementace v praxi může být problematická. V takovém případě je technicky vhodnější i jednodušší využití řešení zvaného LVM.

3.4.2 LVM

LVM je softwarové řešení sloužící ke správě logických disků, a umožňuje vytvoření jednoho či více logických disků nad jedním či více fyzickými disky. Díky tomu je možné sloučit kapacitu více fyzických do jednoho virtuálního disku, což je v běžné terminologii označováno jako RAID 0. Dále je možné využití LVM pro zrcadlení dat na více fyzických disků, čímž jsou chráněna

proti fyzickému poškození libovolného disku. Tato metoda uložení dat je označována jako RAID 1.

Podstatnou funkcionalitou LVM pro tuto práci je možnost vytvoření snímku logického disku, který může být využit ke zlepšení konzistence prováděné zálohy (viz 2.4) bez nutnosti vypnutí virtuálního stroje.

Pro hypervizor je blokové zařízení vytvoření pomocí LVM shodné s tím, které by běžně využil pro připojení fyzického disku do virtuálního stroje, a tudíž pro LVM nemusí implementovat explicitní podporu.

3.4.3 LVMTHIN

LVMTHIN je rozšíření standardní implementace LVM, které při zachování všech jeho vlastností navíc podporuje tzv. thin provisioning, který umožňuje vytvořit logické disky o větší kapacitě, než je celková kapacita fyzických disků. Tato funkcionalita pracuje s předpokladem, že kapacita logického disku není vždy plně využita, a vzniká tím nevyužitelný prostor na disku. Proto je v případě jejího využití fyzický diskový prostor alokován až ve chvíli, kdy na něj skutečně proběhne zápis dat. Ve výsledku se jedná o velmi podobnou funkcionalitu jako u diskových formátů ukládaných v souborech, které implementují dynamickou velikost disku, jako např. VDI, QCOW2, VMDK či VHDX (viz dále).

3.4.4 DRBD

DRBD (Distributed Replicated Block Device) [18] je software implementující distribuované síťové úložiště, které podobně jako LVM vytváří logickou vrstvu nad fyzickými disky, avšak navíc umožňuje asynchronní či synchronní replikaci dat v reálném čase na více serverů. Jedná se tudíž o síťovou variantu RAID 1, který je možné vytvořit napříč více servery. Toho je využíváno především pro serverové clustery s vysokou dostupností, kde výpadek jednoho serveru nevede výpadku provozované služby.

Ačkoliv je možné toto úložiště, podobně jako LVM, používat s libovolným hypervizorem, který explicitně nemusí implementovat podporu pro DRBD, tak byl vyvinut nástroj LINSTOR, který cílí na jednodušší správu úložiště a jeho integraci přímo do hypervizoru.

3.4.5 LINSTOR

LINSTOR je nástroj, který byl vyvinut primárně pro snadnější konfiguraci úložiště DRBD v rozsáhlé infrastruktuře. Přesto je ho v dnešní době možné využít i přímo pro správu samotného LVM bez využití DRBD.

Podstatnou výhodou tohoto nástroje je jeho integrace do virtualizačních platforem, a především Proxmox VE, kterým se tato práce zabývá. Tím je umožněno automatické vytváření logických disků přímo z webové administrace Proxmoxu VE, a není potřeba tyto disky konfigurovat manuálně z příkazové řádky jako by tomu bylo při využití samotného DRBD.

Dále je díky tomuto nástroji možné vytvářet snímky disku, čehož je na pozadí dosaženo za pomoci popisovaného LVM nebo alternativně pomocí souborového systému ZFS. Tato funkcionalita může být opět využita ke zlepšení konzistence prováděné zálohy disku.

Pokud je požadována možnost připojení úložiště na server bez replikace dat na lokálním fyzickém disku, tak je vyžadováno využití úložiště DRBD ve verzi 9, které také nativně podporuje sdílení dat na více než 2 servery. Tuto limitaci bylo dříve nutné obejít za pomoci přidání další vrstvy úložiště, která umožnila replikaci dat na třetí server.

3.5 Dostupná řešení pro virtualizaci stroje

Pro běh virtuálních strojů v domácím prostředí jsou běžně využívány obecně známé hypervizory Oracle VM VirtualBox a Parallels Desktop, které mají jednoduché administrační rozhraní. Avšak vzhledem k nedostatku pokročilých funkcí a nízké podpoře v aplikacích třetích stran nejsou tyto hypervizory vhodné k provozu v produkčním prostředí, kde jsou z těchto důvodů používány pokročilé hypervizory VMware ESXi, Hyper-V, QEMU/KVM nebo Proxmox Virtual Environment.

3.5.1 Oracle VM VirtualBox

Oracle VM VirtualBox [17] (dále jen VirtualBox) je nekomerční virtualizační software, který vyniká především svou jednoduchostí. Z tohoto důvodu je VirtualBox často využíván k virtualizaci v domácích nebo školních podmínkách, kde nejsou vyžadovány pokročilé funkce.

Diskové formáty

VirtualBox podporuje diskové formáty VDI (Virtual Disk Image) [19], VMDK (viz 3.5.6), VHD (viz 3.5.5) a HDD (pouze stará verze z Parallels 2, viz 3.5.2), QCOW (QEMU Copy On Write) [20] a QED (QEMU Enhanced Disk) [21]. Podpora vytváření snímků je implementována pro všechny zmíněné formáty.

- **VDI** - Nativní formát, který podporuje fixní i dynamickou velikost disku.

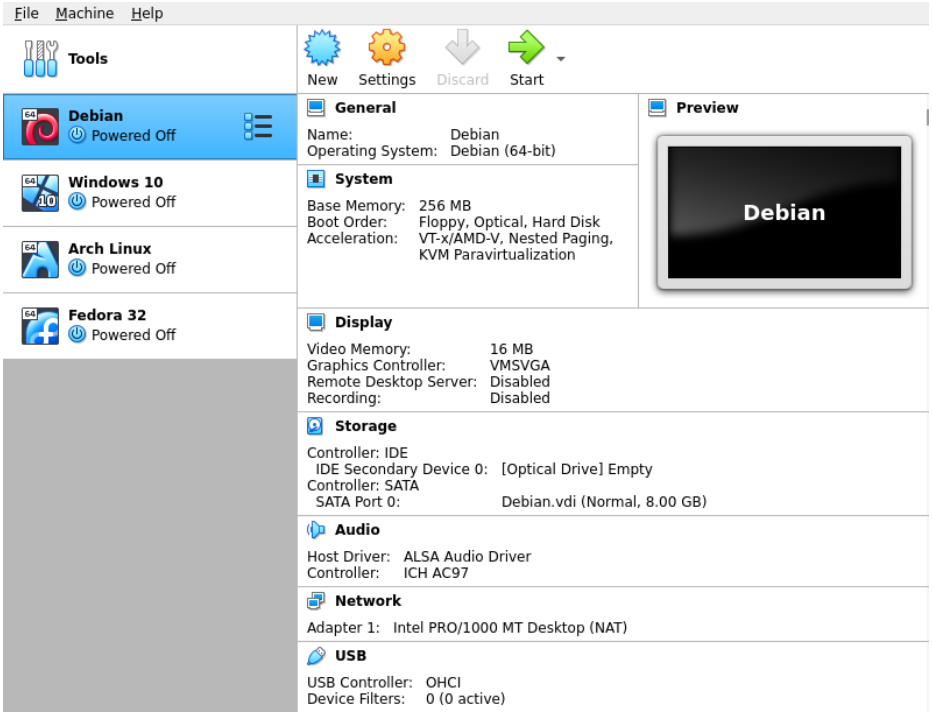
- **QCOW** - Předchůdce formátu QCOW2 (viz 3.5.3), který již v současné době není QEMU podporován, a proto není jeho používání doporučeno. Podpora tohoto formátu není oficiálně zdokumentována, přesto je stále dostupné jeho zvolení přímo v průvodci vytvoření virtuálního disku.
- **QED** - QED je formát založený na QCOW2, a jeho účelem bylo zvýšení výkonu prostřednictvím odstranění některých funkcionalit. Nakonec vývojáři došli k závěru, že vyšší výkon byl dán především jeho efektivnější implementací, a proto byly tyto změny provedeny přímo ve formátu QCOW2, což vedlo k ukončení vývoje QED. Z tohoto důvodu není doporučeno využívání ani tohoto formátu.

Běhové prostředí

VirtualBox je hypervizor druhého typu, a tudíž je vyžadována jeho instalace na hostitelský operační systém. Mezi podporované hostitelské operační systémy patří Microsoft Windows, GNU/Linux a MacOS. [17]

Administrační prostředí

Virtuální stroje je možné spravovat z příkazové řádky nebo prostřednictvím desktopové aplikace (obr. 3.4). Vzdálená správa hypervizoru není nativně podporována, a je nutné tuto problematiku řešit alternativně (např. pomocí vzdálené plochy).



Obrázek 3.4: Administrační rozhraní Oracle VM VirtualBox

3.5.2 Parallels Desktop

Parallels Desktop [22] je komerční virtualizační software, který je podobně jako VirtualBox primárně určen k virtualizaci v domácím prostředí.

Diskové formáty

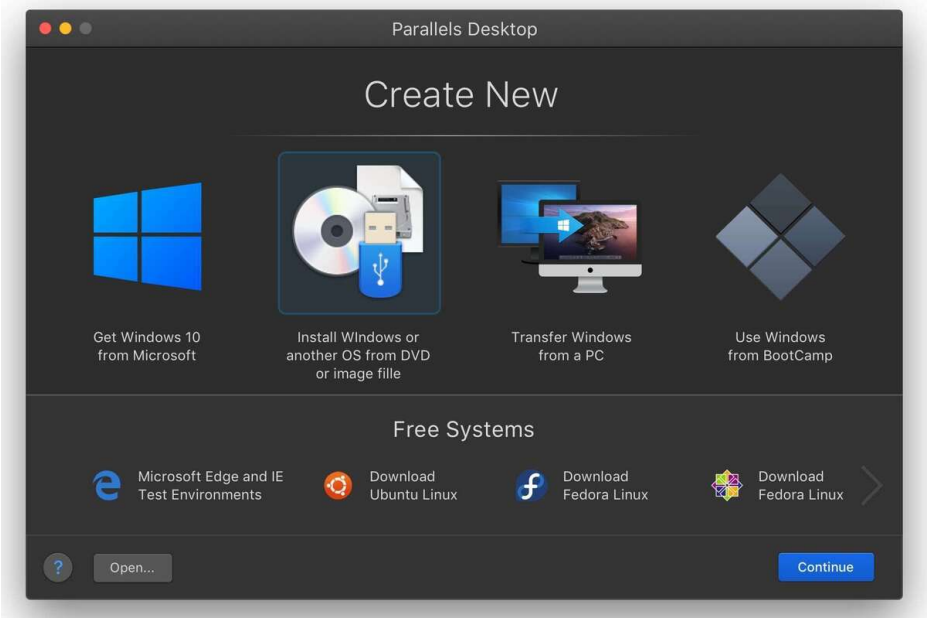
Parallels implementuje podporu pouze pro vlastní diskový formát HDD (Hard Disk), který podporuje fixní i dynamickou velikost virtuálního disku a vytváření jeho snímků.

Běhové prostředí

Stejně jako v případě VirtualBoxu se jedná o hypervizor druhého typu. Podporovanými hostitelskými operačními systémy jsou Microsoft Windows a MacOS.

Administrační prostředí

Parallels je možné spravovat z prostředí příkazové řádky nebo pomocí desktopové aplikace (obr. 3.5) bez možnosti nativní vzdálené správy.



Obrázek 3.5: Administrační rozhraní Parallels Desktop [23]

3.5.3 QEMU

QEMU (Quick Emulator) je nekomerční hypervizor druhého typu s otevřeným zdrojovým kódem pod licencí GNU GPLv2 [24]. Tento hypervizor krom virtualizace běžných hardwarových prostředků umožňuje emulaci procesorů jiných architektur, než je architektura hostitelského stroje. Tato emulace probíhá za pomoci dynamického binárního překladu, což s sebou přináší vysoké nároky na procesor. Avšak je díky tomu umožněna virtualizace stroje na platformě ARM na hostitelském stroji s architekturou x86.

Diskové formáty

QEMU implementuje podporu pro diskové formáty RAW, QCOW2 a VMDK.

- **RAW** - Diskový formát RAW je jednoduchý formát, který obsahuje pouze obraz virtuálního disku bez dalších metadat. Díky tomu je umožněno jeho snadné připojení pomocí standardních nástrojů dostupných v systému GNU/Linux. Nevýhodou je, že QEMU pro tento formát v základu neimplementuje žádné pokročilé funkce jako např. snímky nebo podporu persistentních dirty bitmap, které jsou potřeba pro účely inkrementální zálohy. Nicméně díky jednoduchosti tohoto formátu je možné přidání dalších vrstev, které jsou schopné tuto funkcionalitu dodat. Jedním z příkladů je LVM, díky kterému je možné vytvářet snímky disku v tomto formátu.

- **QCOW2** - Jedná se o pokročilý diskový formát, který podporuje kompresi a šifrování dat, vytváření snímků, dynamickou velikost souboru s diskem i ukládání libovolných metadat, mezi které patří persistentní dirty bitmapy.
- **VMDK** - Jedná se o svobodnou reimplementaci diskového formátu VMDK, který původně pochází z platformy VMware (viz 3.5.6).

Běhové prostředí

QEMU podporuje běh na operačních systémech GNU/Linux, Microsoft Windows i MacOS [25]. Problém výkonu softwarové emulace QEMU řeší pomocí tzv. akcelerátorů, které výrazně zvyšují výkon virtualizace.

Jelikož se jedná o hypervizor druhého typu, tak nejsou kladeny žádné požadavky na fyzický hardware, a jeho podpora je přenesena na hostitelský stroj.

Administrační rozhraní

QEMU je zamýšleno především jako virtualizační nástroj, který je ovládán z prostředí příkazové řádky. Není zde žádná výchozí možnost jak spravovat konfiguraci virtuálních strojů, jelikož se tato konfigurace předává pomocí parametrů příkazové řádky při spuštění QEMU, a není nikde uložena. Virtuální stroj může být za běhu spravován pomocí QAPI (QMP API [26]), přičemž se primárně jedná o síťové rozhraní využívající protokol QMP (QEMU Monitor Protocol).

Na QEMU staví široké množství platforem, jako např. libvirt [27] nebo Proxmox VE, které mimo jiné řeší problematiku persistentní konfigurace virtuálních strojů i jejich správu pomocí vlastních rozhraní, které jsou obvykle ve formě příkazové řádky nebo webové administrace. [4]

3.5.4 KVM

KVM neboli Kernel-based Virtual Machine [1] je nekomerční jaderný modul s otevřeným zdrojovým kódem, který umožňuje přepnout linuxové jádro do režimu hypervizoru prvního typu. V tomto případě se každý virtuální stroj stane procesem na hostitelském stroji, což umožňuje využívat standardních komponent linuxového jádra.

Ačkoliv samotné KVM lze využít k provozu virtuálních strojů, tak nebývá k tomuto účelu samo o sobě využíváno, jelikož se jedná především o hardwarově akcelerované izolované prostředí, které lze kromě provozu virtuálních

strojů využít i ke spuštění libovolného kódu. Pro kompletní ovládání slouží pouze KVM API [28], které je dostupné skrz znakové zařízení `/dev/kvm`. Jeho přímé využívání je výrazně komplikované, jelikož je k jeho použití vyžadována znalost programování, a tudíž slouží především pro zpřístupnění hardwarově akcelerované virtualizace dalším projektům. Jedním z nejpoužívanějších projektů je např. popisované QEMU, které využívá KVM jako akcelerátor, pokud je hardwarová platforma hostovaného a hostitelského stroje shodná.

Běhové prostředí

Jelikož je KVM pouhým modulem linuxového jádra, tak je pro jeho běh vyžadován operační systém Linux. Dále je vyžadován procesor Intel, AMD, ARM, PowerPC nebo S390 s podporou hardwarové virtualizace. Minimální požadavky na výkon a hardwarové zdroje nejsou oficiálně uvedeny, a můžou se lišit podle použité linuxové distribuce. [29]

3.5.5 Hyper-V

Hyper-V je virtualizační platforma vyvíjená společností Microsoft [30], a jedná se o hypervizor prvního typu, který bývá mylně označován za hypervizor druhého typu. Důvodem je, že inicializace hypervizoru a následná správa probíhá z existující instalace operačního systému Microsoft Windows [31], který se po zapnutí podpory Hyper-V přepne do režimu privilegovaného virtuálního stroje (hostitelské domény).

Virtualizační platforma Hyper-V je poskytována zdarma v rámci licence pro operační systém Microsoft Windows ve verzích Pro, Enterprise, Education a Server. Zároveň tuto licenci nelze uplatnit pro případné virtuální stroje s Microsoft Windows, a je potřeba její pořízení pro každý systém zvlášť.

Diskové formáty

Hyper-V implementuje podporu pro vlastní formáty VHD (Virtual Hard Disk) a VHDX (význam přidaného písmene X není definován).

- **VHD** - VHD je starší diskový formát, který podporuje pouze fixní velikost disku, a tudíž zabírá svou plnou velikost již po vytvoření. Dále není implementována podpora pro ukládání vlastních metadat ani ochrana proti poškození dat. Maximální velikost disku je omezena na 2 TB.

- **VHDX** - Formát VHDX je nástupce staršího formátu VHD, který již implementuje podporu pro ukládání metadat, ochranu proti poškození uložených dat a dynamickou velikost souboru s diskem. Zároveň došlo k navýšení maximální velikosti disku na 64 TB.

Vytvoření snímku je podporováno formátem VHD i VHDX. V takovém případě je vytvořen tzv. diferenciální (rozdílový) disk ve formátu AVHD či AVHDX, kam jsou zapisovány nové změny.

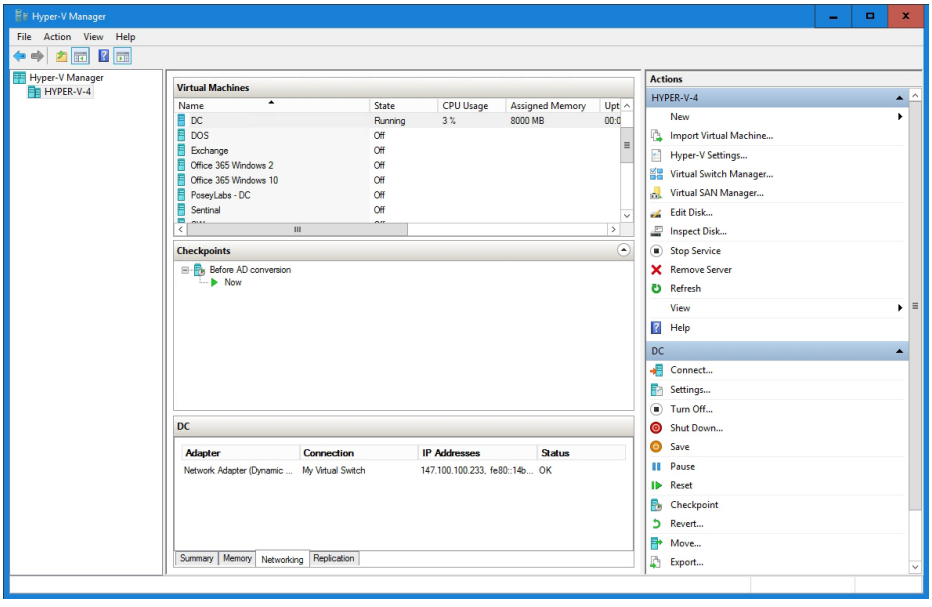
Běhové prostředí

Jelikož se jedná o hypervizor prvního typu, tak je běhovým prostředím samotný hardware. Minimálními požadavky jsou 64-bit CPU s podporou SLAT (Second Level Address Translation), hardwarové virtualizace, hardwarově vynucené prevence spuštění dat (DEP neboli Data Execution Prevention) a paměť RAM o minimální velikosti 4 GB.

Hyper-V podporuje na hostovaném stroji běh operačních systémů Microsoft Windows, Linux a FreeBSD. [32]

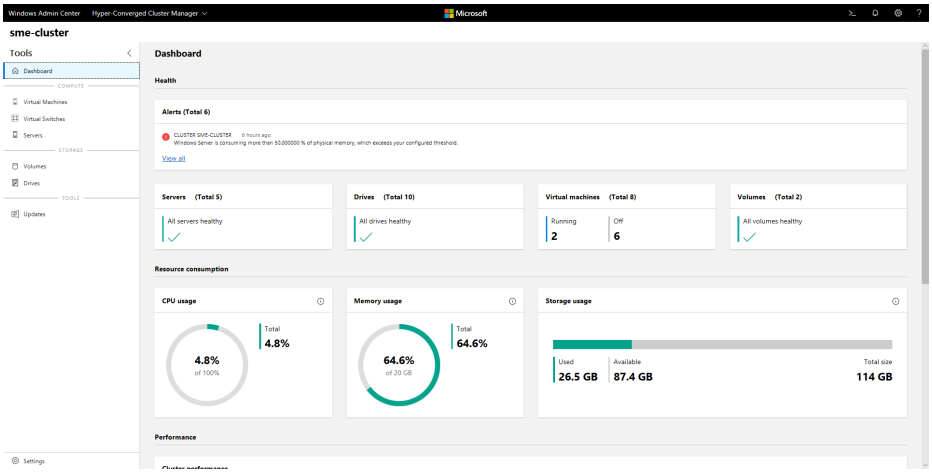
Administrační rozhraní

Oficiálním rozhraním pro správu je Hyper-V Manager (obr. 3.6), který je dostupný jako předinstalovaná aplikace na hostitelském systému. Podporuje správu virtuálních strojů, živou migraci na jiný hypervizor v clusteru i monitoring využití hardwarových prostředků.



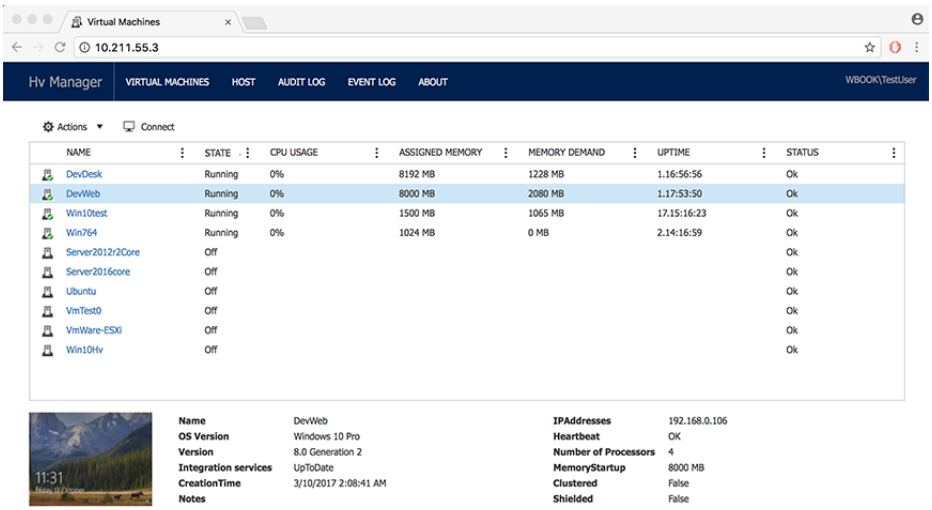
Obrázek 3.6: Administrační rozhraní Hyper-V Manager [33]

Dále je možné využít dodatečně instalované rozhraní Windows Admin Center [34], které je obsluhováno z prostředí webového prohlížeče. Tento nový nástroj není jen nástupcem zmiňovaného Hyper-V Manageru, ale je náhradou pro původní desktopové nástroje, které byly využívány pro správu firewallu, služeb, úložiště, registrů, atd.



Obrázek 3.7: Administrační rozhraní Windows Admin Center [34]

K dispozici je také neoficiální webové rozhraní pod názvem HV Manager [35], které slouží primárně pro monitoring a základní ovládání virtuálních strojů (např. vypnutí, zapnutí, restart). Nejužitečnější funkcí HV Manageru je možnost ovládat virtuálnímu stroj pomocí vzdálené plochy.



Obrázek 3.8: Administrační rozhraní HV Manager [35]

3.5.6 VMware ESXi

VMware ESXi (Elastic Sky X Integrated) [15] je jedním z nejpokročilejších hypervizorů prvního typu používaných v podnikové sféře. Tento hypervizor je postaven na modifikovaném linuxovém jádře, které je rozšířeno o proprietární virtualizační komponenty. Jednou z nich je vmkernel, který je implementován architekturou mikrokernelu, a jeho účelem je přidělování přístupu k procesoru, paměti a dalším fyzickým prostředkům.

Běhové prostředí

Jedná se o hypervizor prvního typu, díky čemuž nedochází k instalaci hostitelského operačního systému, a běhovým prostředím je fyzický hardware. Podmínkou instalace VMware ESXi je 64-bit CPU na architektuře x86 nebo ARM s nejméně dvěma jádry. Pro možnost vytváření 64-bit virtuálních strojů je vyžadován procesor s podporou technologie Intel VT-x nebo AMD-V. [14, 36]

VMware ESXi dále musí implementovat podporu pro použitý hardware, kterou je možné ověřit pomocí oficiálního nástroje VMware Compatibility Guide. [37]

Diskové formáty

Jediným podporovaným diskovým formátem je proprietární VMDK (Virtual Machine Disk). Tento formát implementuje podporu pro vytváření snímků a dynamickou velikost disku, díky které virtuální disk nezabírá na souborovém systému svoji plnou velikost ihned po jeho vytvoření, ale postupně roste až při zápisu dat. Dále je možné soubor s diskem rozdělit na více souborů o maximální velikosti 2GB, čehož může být využito v případě, že velikost virtuálního disku překročí maximální možnou velikost souboru na souborovém systému hostitele. [38]

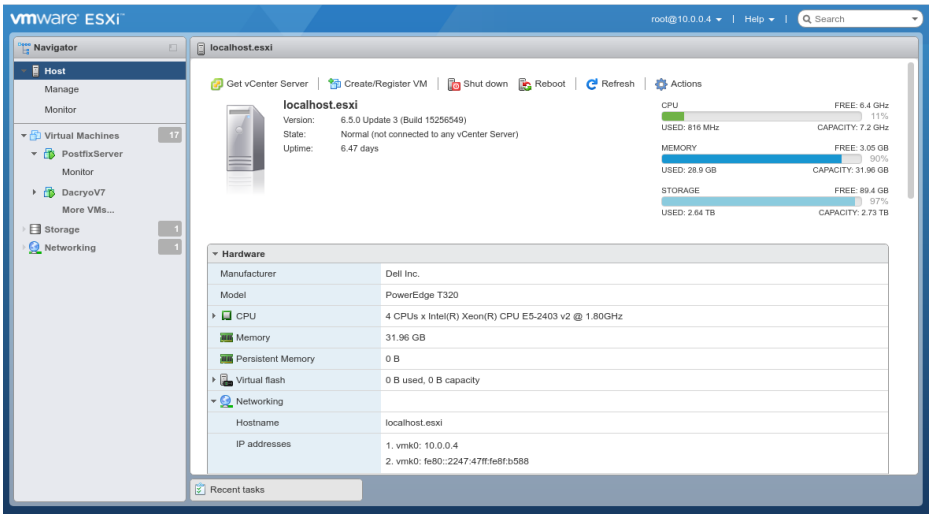
Administrační rozhraní

VMware ESXi poskytuje rozhraní DCUI (Direct Console User Interface), který umožňuje např. základní konfiguraci sítě, zabezpečení, ESXi shellu a vzdáleného přístupu pomocí SSH. Skrz ESXi shell je možné provádět konfiguraci hostitelského stroje na vyšší úrovni, což zahrnuje instalaci balíčků, konfiguraci hardwaru, správu úložiště nebo základní ovládání již vytvořených virtuálních strojů. [39]

Nejvyšší úroveň správy hypervizoru je umožněna skrz webové administrační rozhraní (obr. 3.9), které slouží ke kompletní správě virtuálních strojů,

virtuálních sítí, úložiště, atd. Dále jsou v administraci dostupné jednoduché monitorovací nástroje, které umožňují sledování využití jednotlivých hardwarových prostředků.

VMware také poskytuje VMware vCenter Server, který umožňuje centrální správu několika ESXi hostů. Umožňuje například migraci virtuálních strojů (s produktem vMotion [40] i za běhu), jednotné přihlašování (SSO) [41], centrální aktualizaci ESXi atd. [42]



Obrázek 3.9: Administrační rozhraní VMware ESXi

3.5.7 Proxmox Virtual Environment

Podobně jako VMware ESXi, je i Proxmox Virtual Environment (zkráceně Proxmox VE) kompletní řešení pro provoz a správu virtuálních strojů. Na rozdíl od ostatních popsaných virtualizačních řešení se nejedná o projekt, který řeší problematiku virtualizace od samotného počátku, ale využívá upravenou verzi QEMU (viz 3.5.3) s akcelerátorem KVM (viz 3.5.4). Jako hostitelský operační systém je využívána linuxová distribuce Debian [43].

Diskové formáty

Jelikož platforma Proxmox VE plně využívá virtualizačního řešení QEMU-/KVM, tak podporuje shodné diskové formáty RAW, QCOW2 i VMDK (viz 3.5.3).

Běhové prostředí

Proxmox VE je distribuován ve formě instalačního obrazu, ze kterého je instalován přímo na hardware hostitelského stroje. Krom toho je Proxmox

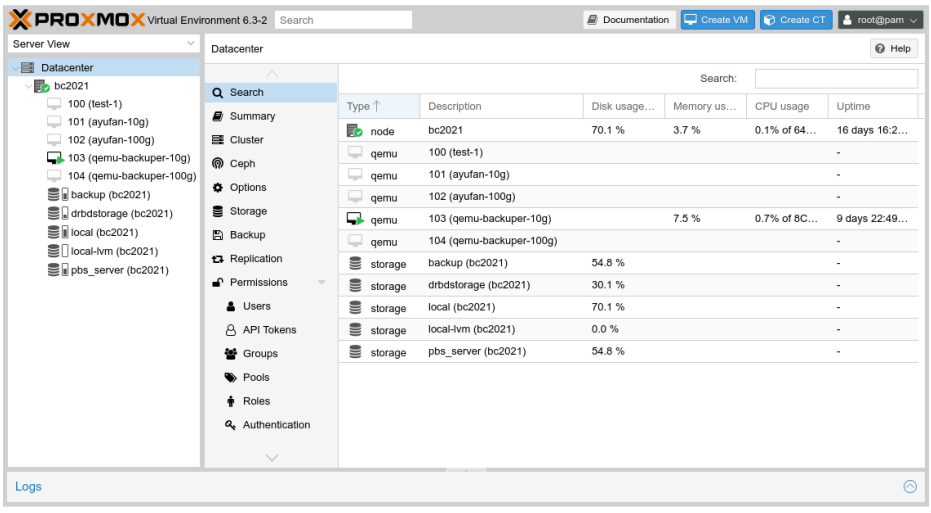
VE možné nainstalovat na existující instalaci operačního systému Debian.

Doporučenými požadavky jsou 64-bitový procesor Intel EMT64 nebo AMD64 s podporou Intel VT nebo AMD-V z důvodu využití plné virtualizace pomocí KVM a alespoň 2 GB RAM. Pro vyšší výkon a spolehlivost je doporučené rychlé redundantní úložiště, a redundantní síťová karta o rychlosti alespoň 1 Gbps. Hostitelský stroj s horšími parametry je doporučen pouze pro testovací účely. [44]

Administrační prostředí

Primárním administračním rozhraním pro Proxmox VE je webová aplikace (obr. 3.10), která je dostupná z libovolného webového prohlížeče podporujícího HTML5 (doporučeny jsou Firefox [45] nebo Google Chrome [46]). Webová administrace umožňuje kompletní správu datového úložiště, virtuálních strojů, migraci virtuální strojů napříč clusterem, správu uživatelů, a interaktivní přístup k virtuálním strojům za pomoci konzole. Alternativně je také možné přistupovat přímo k virtuálním strojům s využitím protokolu SPICE (Simple Protocol for Independent Computing Environments) [47], který vyžaduje přidání virtuálních zařízení zajišťující přenos obrazu a zvuku, a instalaci SPICE klienta (doporučený je virt-viewer [48]) na stroj administrátora.

V případě nemožnosti využití grafického rozhraní je dostupná sada nástrojů pro příkazovou řádku hostitelského stroje, díky čemuž je umožněna jednoduchá automatizace pomocí skriptovacích jazyků.



Obrázek 3.10: Administrační rozhraní Proxmox Virtual Environment

3.6 Shrnutí

Výběr vhodného hypervizoru závisí především na požadovaných funkcionalitách a možnosti integrace s dalšími produkty, v čemž dominují Hyper-V a VMware ESXi, pro které je dostupná široká škála pokročilých zálohovacích řešení (viz dále).

Oracle VM VirtualBox	Parallels Desktop	QEMU	KVM	Proxmox VE	Hyper-V	VMware ESXi
Typ hypervizoru						
druhý	druhý	druhý	první	první	první	první
Běhové prostředí						
Windows, Linux, MacOS	Windows, MacOS	Windows, Linux, MacOS	Linux	Debian	Windows	Fyzický stroj bez OS
Webová administrace						
Ne	Ne	Ne	Ne	Ano	Ano	Ano
Živá migrace virtuálního stroje na jiný hypervizor						
Ano	Ne	Ano	-	Ano	Ano	Ano
Podporované formáty disků						
VDI, QCOW, QED, VMDK, HDD (Parallels 2)	HDD	RAW, QCOW2, VMDK	-	RAW, QCOW2, VMDK	VHD, VHDX	VMDK
Dostupný zdarma						
Ano	14-denní zkušební verze	Ano	Ano	Ano	Součástí licence Microsoft Windows	Do 8 vCPU na virtuální stroj

Tabulka 3.1: Souhrn parametrů popsaných typů záloh

Z tabulky 3.1 vyplývá, že dostupnost webové administrace stále není standardem, a ze zkoumaných virtualizačních platforem ji nabízí pouze Proxmox VE, Hyper-V a VMware ESXi. Oproti tomu živá migrace virtuálního stroje již standardem je, a kromě Parallels Desktop ji podporují všechna popisovaná řešení. Co se týče podporovaných formátů disků, tak každé řešení využívá ve výchozí konfiguraci vlastní formát. Avšak formát VMDK od

VMware je velmi rozšířen a jeho podpora je implementována i v ostatních platformách. A to ať už ve formě možnosti jeho přímého připojení k virtuálnímu stroji, nebo alespoň jeho konverze do formátu dané platformy. Všechna popisovaná řešení kromě Parallels Desktop a Hyper-V je možné získat plně zdarma buď s mírnými omezeními nebo úplně bez omezení.

Vyvíjené řešení bude implementovat podporu pro Proxmox Virtual Environment, který je využíván na Fakultě aplikovaných věd Západočeské univerzity v Plzni, jelikož se jedná o rozšířenou platformu s otevřeným zdrojovým kódem a komunitní i placenou podporou. Na základě analýzy dostupných virtualizačních řešení byly stanoveny následující požadavky:

- Možnost využívání vyvinutého řešení s hypervizory QEMU/KVM (pouze z příkazové řádky) i Proxmox Virtual Environment (z webové administrace)
- Zachování možnosti živé migrace virtuálního stroje na jiný hypervizor v rámci clusteru

4 Zálohování virtuálních strojů

Problematika provádění záloh virtuálních strojů je mírně odlišná od zálohy fyzických strojů. Proto je vhodné pro tyto účely zvolit nástroje, které přímo podporují zálohování virtuálních strojů provozovaných pomocí zvolené virtualizační platformy.

4.1 Specifika zálohování virtuálních strojů

Úloha zálohy virtuálního stroje je odlišná od běžné zálohy fyzického disku, který je využíván v kombinaci s fyzickým strojem. Virtuální stroj je možné zálohovat externě z úrovně hypervizoru, který má přístup ke všem jeho diskům. Díky tomu je virtuální stroj možné zálohovat za zapnutého i vypnutého stavu. Lokální disky fyzického stroje tímto způsobem není možné zálohovat a je vždy nutné provést jejich zálohu z běžícího operačního systému, což může vést ke zhoršení konzistence výsledné zálohy.

Dále je možné virtuální stroj pro účely vytvoření zálohy dočasně pozastavit, což vede ke zlepšení konzistence zálohy v případě, že zálohovací řešení potřebuje provést více na sobě závislých operací, kdy by zápis dat na zálohovaný disk vedl k nekonzistenci výsledné zálohy. Pozastavení fyzického stroje není pro účely zálohy možné, jelikož by došlo i k pozastavení průběhu samotné zálohy.

4.2 Dostupná řešení pro zálohování virtuálních strojů

Pro zálohu virtuálních strojů je v současné době dostupné velké množství nástrojů, které podporují zálohu různých virtualizačních platforem. Pro následující srovnání byly zvoleny nejrozšířenější nástroje, které pokrývají všechny popisované virtualizační řešení.

4.2.1 Veeam Backup and Replication

Veeam Backup and Replication [49] je jedním z nejpokročilejších nástrojů, který se problematikou zálohování virtuálních strojů zabývá.

Podporované virtualizační platformy

Podporovanými virtualizačními platformami, které mohou být zálohovány, jsou pouze VMware vSphere [50] a Microsoft Hyper-V, a to bez požadavků na hostovaný operační systém a využívaný souborový systém. Dále jsou podporovány cloudové infrastruktury Microsoft Azure [51] a AWS (Amazon Web Services) [52]. Veeam Backup and Replication také podporuje zálohu fyzických strojů využívající operační systémy Microsoft Windows a Linux, které využívají souborové systémy btrfs, ext 2/3/4 [10], FAT [53], HFS [54], JFS [55], NILFS [56], NTFS [57], ReiserFS [58] nebo XFS [59]. [49, 60]

Možnosti zálohy a obnovy virtuálního stroje

Záloha virtuálního stroje provozovaném v prostředí VMware ESXi i Microsoft Hyper-V využívá podpory snímků, které zajišťují neměnnost dat po celou dobu trvání zálohy. Veeam Backup and Replication také podporuje diferenciální a inkrementální zálohy s rozdílovým přenosem, který využívá funkce sledování změněných dat od poslední zálohy, které obě podporované virtualizační platformy implementují. Za předpokladu, že informace o změně dat nejsou dostupné, tak Veeam za pomoci centrálního zálohovacího serveru získá záložní metadata, která využije pro jejich detekci. [61]

Veeam Backup and Replication podporuje instantní zotavení virtuálního stroje, které vytvoří dočasný virtuální stroj s připojeným úložištěm přímo ze zálohy, čímž je minimalizován čas výpadku za cenu dočasného snížení výkonu. K tomuto účelu je využíván proprietární ovladač, který přesměrovává čtecí operace na zálohovací server. Pro následnou finalizaci obnovy je vyžadování spuštění migrace zálohy do produkčního úložiště. V průběhu této operace proprietární ovladač ví, která data již byla obnovena, a tyto čtecí operace pracují s lokálním úložištěm, které poskytuje plný výkon. Jakmile je obnova kompletně dokončena, tak se ukončí spojení se zálohovacím serverem, a virtuální stroj dále pokračuje v běhu s plným výkonem produkčního úložiště. [62, 63]

Další metodou je kompletní obnova virtuálního stroje, kterou je možno provést do původní nebo jiné lokace. Nevýhodou proti předchozí popisované metodě je nutnost vyčkání na kompletní dokončení obnovy, než je možné virtuální stroj znovu spustit. [64, 65]

Poslední možností je obnova jednotlivých souborů virtuálního stroje. Může se jednat o obnovu souboru s virtuálním diskem (.vmdk, .vhd, atd.) nebo souboru s konfigurací virtuálního stroje. [66, 67]

Licencování

Zálohovací platforma od společnosti Veeam je nabízena především ve formě komerčního produktu. Přesto v nabídce existuje i komunitní edice s omezenými funkcemi, kterou je možné využít až pro 10 fyzických nebo virtuálních strojů. Tato edice je dodávána bez podpory cloudů Microsoft Azure a AWS, API pro využití aplikacemi třetích stran, zálohy ze snímku sekundárního úložiště, monitoringu, orchestrace obnovy po havárii, nebo koncového (end-to-end) šifrování. Jedná se tudíž o licenci určenou pro domácí použití nebo malé firmy, kde nejsou tyto pokročilé funkce vyžadovány. Pro střední a velké firmy jsou nabízeny placené edice standard, enterprise a enterprise plus. [49, 68]

Veeam nabízí i další nástroje ze svého zálohovacího ekosystému, které mohou být zakoupeny zvlášť nebo jako součást univerzální licence (VUL) [69]. Dostupné univerzální licence jsou Veeam Backup Essentials™, Veeam Backup & Replication™ a Veeam Availability Suite™. Všechny zmíněné balíčky nabízí funkcionality z licence enterprise plus, avšak pouze první a třetí obsahuje navíc monitorovací a diagnostický nástroj Veeam One [70]. První zmíněná univerzální licence má také rozsahové omezení na maximálně padesát klasických licencí a minimální nákup pěti licencí. Informace o jejich cenách nejsou veřejně dostupné, a jsou vyměřovány na základě individuální poptávky. [71]

4.2.2 Proxmox Backup Server

Proxmox Backup Server [72] je nejnovější zálohovací řešení od vývojářů Proxmox VE, které slouží k zálohování virtuálních strojů na binární úrovni provozovaných na této platformě, ve které možnost pokročilého zálohování dlouhodobě chyběla, jelikož je ve výchozím stavu v Proxmoxu VE dostupný pouze jednoduchý vestavěný nástroj VZDump [4], který umožňuje pouze vytváření a jednoduchou správu plných záloh.

Součástí řešení je agent Proxmox Backup Client, který podporuje zálohu linuxové distribuce Debian na úrovni souborů.

Podporované virtualizační platformy

Mezi podporované zálohované virtualizační platformy patří prozatím pouze zmíněný Proxmox VE, kde je možné zálohovat virtuální stroje i LXC kontejnery, přičemž podpora pro jiné virtualizační platformy zatím není v plánu. Avšak v budoucnu by měla být přidána podpora zálohy samotného hypervizoru. [72, 73]

Možnosti zálohy a obnovy virtuálního stroje

Proxmox Backup Server podporuje plné i inkrementální zálohy. Pro provedení inkrementální zálohy je vyžadován zapnutý virtuální stroj. Zároveň nesmí být virtuální stroj mezi jednotlivými inkrementálními zálohami vypínán, jelikož dojde ke ztrátě informací o provedených změnách na disku, a je vyžadováno provedení plné zálohy.

Obnova celého virtuálního stroje probíhá ze standardního rozhraní Proxmox VE, a využívá nového nástroje `pbs-restore`, který je v případě potřeby možné využívat přímo z prostředí příkazové řádky. [74]

Licencování

Nástroj je uvolněn pod svobodnou licencí GNU AGPL v3 [75], a je ho možné používat plně zdarma bez omezení nebo s placenými plány community, basic, standard a premium, které navíc nabízí přístup k repozitáři určenému pro produkční prostředí a s výjimkou plánu community i technickou podporu. Samotný licenční model je založen na počtu zálohovacích serverů bez omezení počtu zálohovaných klientů. [76]

4.2.3 Bacula

Bacula [77] je další pokročilé zálohovací řešení, které podporuje zálohu fyzických i virtuálních strojů.

Podporované virtualizační platformy

Zálohovací řešení Bacula podporuje zálohu virtualizačních platforem Hyper-V, VMware ESXi, RHV (Red Hat Virtualization) [78], KVM, Xen a Proxmox VE. Pro každou z platforem jsou dostupné různé možnosti záloh a obnovy dat (viz dále).

Možnosti zálohy a obnovy virtuálního stroje

Pro VMware ESXi je podporována plná, inkrementální a diferenciální záloha na binární úrovni. Zároveň je pro hostované operační systémy Linux a Windows využívající souborové systémy ext3, ext4, XFS, FAT a NTFS dostupná obnova na úrovni souborů za běhu virtuálního stroje.

Pro virtuální stroje provozované pomocí virtualizace KVM, konkrétně na platformě libvirt, je podporováno vytváření plných, inkrementálních a diferenciálních záloh na binární úrovni. Dále je možné vytvářet zálohy na úrovni souborů, které je na pozadí prováděno pomocí nástroje guestfish [79].

Při použití s platformami Hyper-V, Red Hat Virtualization, Xen a Proxmox VE je dostupná pouze plná záloha na binární úrovni bez možnosti obnovy jednotlivých souborů. [80]

Licencování

Bacula je dostupná zdarma ve verzi Community pod svobodnými licencemi GNU GPLv2 a GNU AGLPv3. Druhou dostupnou verzí je komerční licence Enterprise, která zpřístupňuje možnost zálohy virtuálních strojů ve zmíněných virtualizačních platformách, které s licencí Community není možné zálohovat. Tuto komerční licenci je možné bezplatně vyzkoušet v rámci třicetidenní zkušební lhůty. [81]

4.2.4 Acronis Cyber Backup

Posledním ze zkoumaných nástrojů je Acronis Cyber Backup [82], který se svými funkcemi a podporovanými platformami podobá popisovanému Veeam Backup and Replication.

Podporované virtualizační platformy

Acronis Cyber Backup podporuje zálohu virtuálních strojů v prostředí VMware, Hyper-V, KVM, RHEV (Red Hat Enterprise Virtualization), Oracle KVM [83] a Citrix XenServer [84]. Avšak podpora na úrovni hypervizoru je pouze pro VMware a Hyper-V, a pro ostatní virtualizační řešení je vyžadována instalace agenta na hostovaný stroj s operačním systémem Microsoft Windows nebo Linux. Tento způsob implementace podporují i ostatní popisovaná zálohovací řešení, a ze strany agenta zde není vyžadována podpora pro konkrétní hypervizor. Podpora pro tyto řešení je tedy pouze na úrovni licence, kterou není nutné pořizovat pro každý virtuální stroj, ale postačuje jedna licence pro celý hypervizor. [82, 85]

Možnosti zálohy a obnovy virtuálního stroje

Podporovány jsou plné, inkrementální a diferenciální zálohy, které podobně jako u ostatních zálohovacích nástrojů fungují na principu zaznamenávání změn přímo v Hyper-V nebo VMware. Funkci zaznamenávání změn je možné mít vypnutou, přičemž se pak pro zjištění rozdílu využije metoda porovnávání stavu disku s poslední provedenou zálohou, což má velký dopad na rychlost zálohy.

Podobně jako Veeam Backup and Replication (viz 4.2.1), tak i Acronis Cyber Backup podporuje instantní obnovu, která čte data přímo ze zálohovacího serveru, díky čemuž není potřeba čekat na jejich kompletní obnovu do produkčního úložiště. Takto obnovený virtuální stroj není oficiálně doporučeno provozovat déle než tři dny, jelikož dochází k vynucení uchování zálohy, a nemohou být aplikovány pravidla pro mazání starých záloh. Avšak je podporováno jeho převedení na standardní na standardní virtuální stroj s plným výkonem úložiště, a to kompletně bez výpadku. Krom tohoto typu obnovy virtuálního stroje je dostupná i standardní obnova, která vyžaduje vypnutý stroj, a je možné ji provést do původního nebo nového umístění. [86]

Licencování

Zálohovací řešení je komerčním produktem, a je k dispozici zdarma pouze v rámci třicetidenní zkušební lhůty. Pro produkční nasazení je potřeba zakoupení samostatné licence pro každý hypervizor, na kterém je poté možné zálohovat neomezené množství virtuálních strojů. Informace o cenách nejsou veřejně dostupné, a jsou vyměřovány na základě individuální poptávky. [82, 87–89]

4.3 Shrnutí

Přednostně jsou v tabulce 4.1 posuzovány parametry, které přímo souvisí s virtuálními stroji, a záměrně jsou vynechány parametry, které se týkají např. zálohování jednotlivých souborů, které není vázáno na fyzické nebo virtuální prostředí.

Veeam Backup and Replication	Proxmox Backup Server	Bacula Enterprise	Acronis Cyber Backup
Podporované platformy pro zálohy bez agenta			
VMware ESXi a Microsoft Hyper-V	Proxmox VE	Hyper-V, VMware ESXi, RHV, KVM, Xen a Proxmox VE	VMware ESXi a Microsoft Hyper-V
Webová administrace			
Ano	Ano	Ano	Ano
Inkrementální a diferenciální záloha			
Ano	Ano	Pouze VMware ESXi a libvirt	Ano
Inkrementální a diferenciální záloha pomocí sledování změn na disku			
Ano	Pouze bez vypnutí virtuálního stroje	Pouze VMware ESXi a libvirt	Ano
Instantní obnova virtuálního stroje			
Ano	Ne	Ne	Ano
Automatická obnova do jiné lokace			
Ano	Ano	Pouze RHV a Proxmox VE	Ano
Naplánování automatické obnovy v případě havárie			
Veeam Availability Orchestrator	Ne	Ne	Acronis Disaster Recovery Service
Monitoring a reporting			
Veeam ONE	Ano	Ano	Ano
Rozsahové omezení licence zdarma			
do 10 virtuálních strojů	bez omezení	třicetidenní zkušební doba	třicetidenní zkušební doba
Licenční model			
Podle počtu fyzických procesorů na hypervizoru	Podle počtu zálohovacích serverů (všechny funkce jsou dostupné zdarma)	Podle počtu zálohovaných virtuálních strojů	Podle počtu hypervizorů

Tabulka 4.1: Srovnání zálohovacích řešení pro virtuální stroje

Z tabulky vyplývá, že všechny popisované zálohovací řešení implementují

webové rozhraní, které se v dnešní době začíná stávat obecným standardem. Stejně tak bývá stěžejní funkcí podpora inkrementálního a diferenciálního zálohování, které vede k výrazné úspoře diskového prostoru. Oproti tomu automatická obnova v případě havárie stroje je velmi pokročilou funkcionalitou, která není dostupná buď vůbec nebo v rámci dodatečného placeného produktu. Pro virtualizaci QEMU/KVM, kterou se tato práce zabývá, není v současné době dostupné žádné zálohovací řešení podporující automatickou obnovu v případě havárie. Licenční modely často poskytují možnost jak si základní verzi produktu (bez dodatečných služeb a nástrojů) před zakoupením licence vyzkoušet. Případně je možné využívání základní verze produktu po neomezenou dobu v domácích podmínkách (omezení na jednotky zálohovaných zařízení).

Účelem vyvíjeného řešení je implementace vytváření plných a inkrementálních záloh virtuálních strojů provozovaných na platformě Proxmox VE, ale i pomocí samotného QEMU/KVM, pro které v současné době neexistuje řešení, které by v současné době umožňovalo vytváření inkrementálních záloh za pomocí aktivního sledování zápisů dat na disk s využitím dirty bitmapy (viz 2.1.3, 2.2).

5 Implementace

V době psaní práce není dostupný žádný software, který by implementoval podporu pro inkrementální a diferenciální zálohy virtuálních strojů provozovaných na platformě Proxmox VE. Dokonce ani není dostupný žádný software, který by tuto funkcionalitu implementoval alespoň pro QEMU, na kterém Proxmox VE staví, a bylo by ho možné rozšířit.

V průběhu vypracování této bakalářské práce byl sice vydán Proxmox Backup Server, jehož cílem je nabídnout pokročilé zálohovací řešení pro Proxmox VE. Ten ale nepodporuje vytváření inkrementálních záloh za vypnutého stavu virtuálního stroje, a ani po jeho restartu, kdy jsou informace o změněných datech ztraceny, a je nutné provést plnou zálohu.

5.1 Specifikace požadavků

Na základě stanovených požadavků v kapitolách 2.5 a 3.6 budou do vyvíjeného řešení implementovány následující funkcionality:

- Vytvoření a obnova plných a inkrementálních záloh virtuálních strojů provozovaných v Proxmox VE, které využívají virtuální disky ve formátu RAW v kombinaci s úložištěm LINSTOR (viz 3.4.5)
- Možnost správy řešení z webové administrace Proxmox VE, která zahrnuje automatickou konfiguraci nástroje pro virtuální stroje a možnost vytváření, mazání a obnovy záloh
- Možnost samostatného využití nástroje s QEMU/KVM
- Pro účely vytvoření inkrementální zálohy budou změny dat sledovány aktivně již při jejich zápisu, a tato informace bude uchovávána v dirty bitmapě (viz 2.2.1)
- Zachování možnosti živé migrace virtuálního stroje na jiný hypervizor

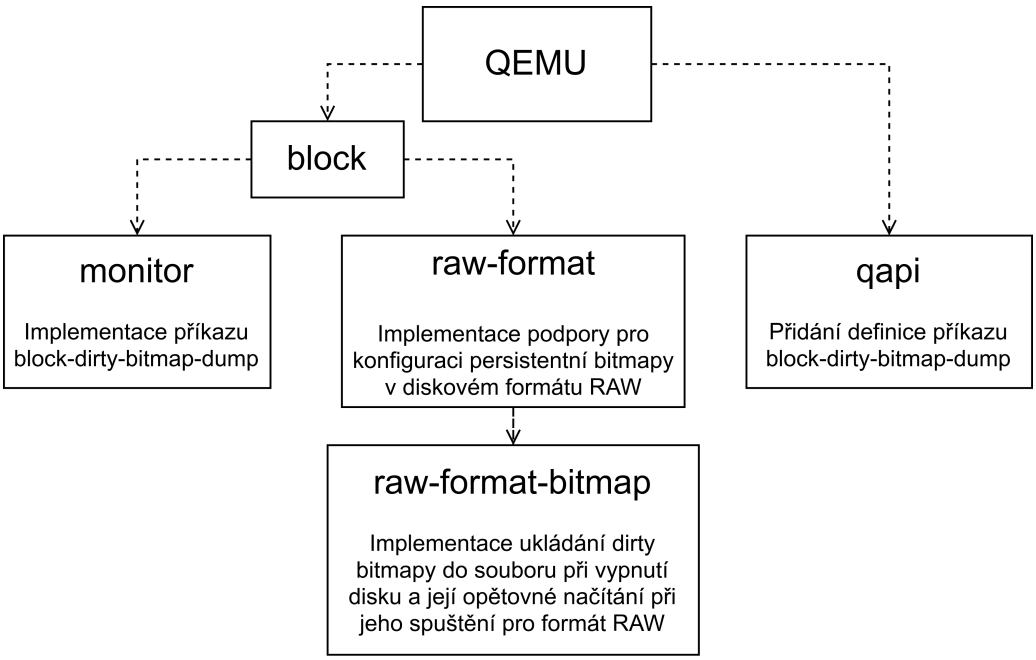
5.2 Struktura řešení

Implementované řešení se skládá ze tří částí. První částí je rozšíření implementace dirty bitmap v projektu QEMU, které jsou aktuálně implementovány pouze pro interní účely, a aplikacím třetích stran není umožněno při-

stupovat k jejich obsahu skrz QMP API. Druhou částí je implementace obslužného nástroje, který bude sloužit pro vytváření a obnovu záloh. Poslední částí je začlenění celého řešení do projektu Proxmox VE a jeho administrace.

5.3 Rozšíření funkce dirty bitmapy v QEMU

Aby bylo možné implementovat další části práce, tak je nutné nejprve rozšířit implementaci dirty bitmap pro diskový formát RAW, který v současné chvíli nepodporuje její persistentní uložení mezi restarty virtuálního stroje. Tím je vynuceno přerušení sekvence inkrementálních nebo diferenciálních záloh po každém vypnutí, a následném spuštění virtuálního stroje. Zároveň je potřeba zpřístupnit obsah dirty bitmapy pomocí QMP API, aby nástroj pro správu mohl získat informace o změně dat, které jsou určeny k záloze.



Obrázek 5.1: Struktura provedených změn v projektu QEMU

5.3.1 Současná implementace dirty bitmap v QEMU

Současná implementace dirty bitmap v QEMU podporuje jejich persistenci pouze pro diskový formát QCOW2, který je pro tuto práci nevyhovující z důvodu problematického nasazení v kombinaci s úložištěm LINSTOR. Přístup k obsahu bitmapy je v současné době možný jen pomocí protokolu NBD (network block device) [90], avšak tento postup není součástí oficiální uživatelské dokumentace, a je výrazně komplikovanější než využití QMP API.

5.3.2 Zpřístupnění dirty bitmap pro externí nástroje

Pro účely poskytnutí přístupu k obsahu bitmapy externím nástrojům bylo QMP API rozšířeno o příkaz `block-dirty-bitmap-dump`, který slouží k získání obsahu dirty bitmapy zakódovaného ve formátu base64. Tento formát je zvolen z důvodu, že QMP API pro svou komunikaci využívá formát JSON, který nepodporuje binární data.

Pro implementaci příkazu bylo potřeba nejprve vytvořit jeho definici. Tyto definice QEMU spravuje pomocí souboru `qapi/block-core.json` ve formátu JSON, který umožňuje specifikovat vstupní a výstupní hodnoty jednotlivých příkazů, a zároveň definovat datové struktury, což umožňuje jejich opakované používání napříč různými příkazy. Na základě tohoto souboru je příkazem `make` vygenerován zdrojový kód v jazyce C, který zajišťuje čtení a syntaktickou analýzu (parsing) spouštěných příkazů. Tento postup eliminuje nutnost psaní vlastního syntaktického analyzátoru pro jednotlivé příkazy a použité datové struktury.

Druhá část implementace příkazu zahrnovala napsání funkce v souboru `block/monitor/bitmap-qmp-cmds.c`, ve kterém je nejprve získán exkluzivní přístup k asynchronnímu I/O, čímž je na velmi krátkou dobu virtuálnímu stroji zamezen přístup na disk, jelikož by informace o případném zápisu dat mohla být ztracena, což by vedlo k poškození všech následujících inkrementálních záloh. Dále v rámci této funkce dojde k exportu a vynulování dirty bitmapy, po kterém následuje uvolnění zámku k asynchronnímu I/O, čímž je virtuálnímu stroji obnoven přístup k disku. V poslední části funkce dojde k serializaci obsahu bitmapy do base64 a naplnění výstupní struktury, která je následně serializována do formátu JSON a odeslána pomocí QMP API.

Požadavek na vynulování dirty bitmapy při jejím exportu je možné nastavovat parametrem `clear`. Díky této možnosti je možné v budoucnu rozšířit práci o podporu diferenciálních záloh bez dalšího zásahu do zdrojového kódu QEMU. K tomuto účelu je sice v projektu QEMU již implementován příkaz `block-dirty-bitmap-clear`, který umožňuje vynulování bitmapy. Avšak ho nebylo možné využít z důvodu, že by virtuální stroj mohl provést zápis na disk mezi voláním příkazu `block-dirty-bitmap-dump` a `block-dirty-bitmap-clear`, což by vedlo k poškození všech následujících inkrementálních záloh.

5.3.3 Persistence dirty bitmapy pro formát RAW

Diskový formát RAW neumožňuje uložení dalších metadat, a tak v něm není možné uložit dirty bitmapy jako u formátu QCOW2, což je nezbytné pro vy-

tvoření nepřerušené sekvence inkrementálních záloh po restartu virtuálního stroje. Z tohoto důvodu bylo zvoleno řešení, kdy jsou dirty bitmapy uloženy v samostatných souborech na hostitelském stroji.

Při spuštění a vypnutí virtuálního disku jsou abstraktní implementací úložiště v QEMU zavolány funkce `bdrv_open` a `bdrv_close` z implementace formátu RAW v souboru `block/raw-format.c`. První z funkcí je využita pro načtení obsahu dirty bitmapy ze souboru. Druhá pak slouží k jejímu opětovnému uložení do souboru. Zdrojový kód pro práci s persistentními dirty bitmapami v kombinaci s formátem RAW je po vzoru formátu QCOW2 umístěn v souboru `block/raw-format-bitmap.c`. Tento soubor obsahuje implementaci samotného uložení resp. načtení persistentní dirty bitmapy a ověření zápisových práv pro uložení bitmapy, které probíhá před zapnutím virtuálního stroje.

V souboru s obsahem dirty bitmap není uložena její konfigurace, která zahrnuje informaci o zvolené granularitě nebo názvu. A to z důvodu, že QEMU nevyužívá žádné konfigurační soubory, skrz které by bylo možné předat cestu k persistentní dirty bitmapě, a tudíž by nemělo informaci ani o jejich existenci. Proto je konfigurace dirty bitmapy předávána pomocí parametrů příkazové řádky v rámci definice disku.

```
dirty-bitmaps.0.filename=/bitmaps/test.bitmap,
dirty-bitmaps.0.granularity=65536,
dirty-bitmaps.0.persistent=on
```

Ukázka kódu 5.1: Konfigurace bitmapy pomocí parametrů QEMU

Každý parametr je rozdělen tečkou do tří částí. První část `dirty-bitmaps` určuje, že se jedná o konfiguraci dirty bitmapy. Druhá část `0` definuje index bitmapy, a který musí být číslován od nuly. Přestože vyvinutý nástroj využívá pro účely vytváření inkrementálních pouze dirty bitmapu s indexem nula, tak je pomocí odlišných indexů možné nadefinovat další dirty bitmapy, které mohou být využívány dalšími aplikacemi např. pro účely monitoringu zápisu dat. Poslední částí je název parametru konkrétní dirty bitmapy.

Parametr `filename` určuje cestu, kde bude dirty bitmapa uložena na hostitelském systému, a je zároveň použit jako její název. K vytvoření souboru s dirty bitmapou dochází automaticky při prvním vypnutí virtuálního stroje, avšak k vytvoření nadřazené složky nedochází, a musí být vytvořena manuálně. Parametr `granularity` slouží k definování granularity dirty bitmapy (viz 2.2.1). Poslední parametr `persistent` je určen pro zapnutí či vypnutí persistence dirty bitmapy pomocí hodnot `on` a `off`. Pokud bude persistence vypnuta, tak nebude dirty bitmapa uložena do zvoleného souboru, a bude znemožněno vytvoření inkrementální zálohy po vypnutí virtuálního stroje.

5.4 Nástroj pro provádění záloh

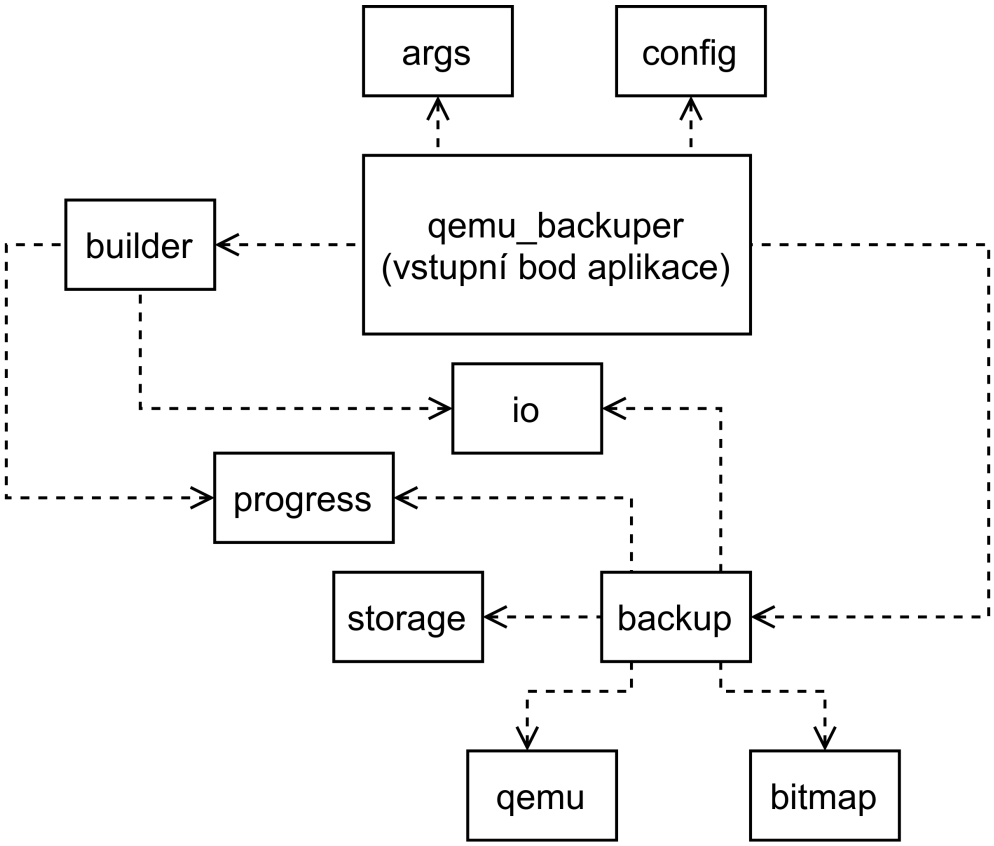
Na základě stanovených cílů v kapitole 2.5 bude tento nástroj implementovat podporu pro vytváření a obnovu plných a inkrementálních záloh virtuálních strojů, jejichž disky využívají úložiště LINSTOR. K detekci změněných dat bude pro účely vytvoření inkrementální zálohy využita technika aktivního sledování zápisů na blokové zařízení a poznamenávání těchto zápisů do dirty bitmapy (viz 2.2).

5.4.1 Zvolené technologie

Pro implementaci nástroje byl zvolen jazyk Python ve verzi 3.7. Hlavním důvodem pro výběr tohoto jazyka je možnost rychlého vývoje, což umožnilo snadnou implementaci prototypu, který mohl být následně rozšířen do plnohodnotné aplikace. Dalším argumentem byl fakt, že je interpret Pythonu 3.7 předinstalován v aktuálně nejnovějším Proxmoxu VE 6.3-1.

5.4.2 Struktura aplikace

Struktura aplikace je členěna do několika samostatných modulů (obr. 5.2), čímž je zajištěna univerzálnost a opakovatelná použitelnost jednotlivých tříd a funkcí, které jsou v těchto modulech implementovány.



Obrázek 5.2: Diagram modulů nástroje pro vytváření a obnovu záloh

args

Pomocí modulu `argparse`, který je součástí standardní knihovny Pythonu, je implementován parsing a validace definovaných argumentů příkazové řádky. Zároveň tento modul zajišťuje automatické generování textové nápovědy při použití přepínače `-h` či `--help`.

bitmap

Obsahem je jediná třída `DirtyBitmap`, jejíž úkolem je namapovat vyexportovanou bitmapu o známé granularitě na skutečnou velikost virtuálního disku. Vytvořený objekt je díky tomu schopný simulovat shodné chování jako by byla granularita dirty bitmapy jedna, čehož je využito v modulu `io` (viz dále).

backup

Tento modul implementuje samotné provedení plné a inkrementální zálohy. Vstupním bodem je funkce `do_backup`, která pro každý zálohovaný disk vy-

tvoří samostatné vlákno funkce `__do_backup_thread`. Paralelizací operace dojde k vytvoření snímků všech disků v co nejkratším možném čase, čímž je zlepšena konzistence zálohy napříč zálohovanými disky.

builder

Modul `builder` je využíván k sestavení disku z plných a inkrementálních záloh. Toto sestavení probíhá virtuálně bez nutnosti ukládání dat na disk za pomocí třídy `ChunkStream`, která implementuje shodné metody jako instance objektu otevřeného souboru, čímž je zajištěna její znovupoužitelnost i s knihovnami třetích stran. Tato třída pracuje s polem instancí objektu `Chunk`, který definuje rozsah obsažených dat v souboru se zálohou. Při čtení z instance objektu `ChunkStream` je následně čteno z takové instance objektu `Chunk`, která obsahuje data na požadované pozici. V případě, že požadovanou část dat obsahuje více objektů, tak je vybrán ten s vyšší prioritou.

Princip funkce je vizualizován na obr. 5.4, kde jednotlivé vrstvy reprezentují instance objektu `Chunk` a všechny tyto vrstvy tvoří dohromady jednu instanci objektu `ChunkStream`.

io

Modul sloužící k obecnému kopírování dat mezi dvěma instancemi otevřeného souboru. Jak již bylo zmíněno, tak zdrojem kopírovaných dat může být i instance objektu `ChunkStream`. Dále se zde nachází obecná implementace pro sdělování průběhu operace (viz dále) a možnost kopie vybraných částí dat na základě předané instance objektu `DirtyBitmap`.

storage

V tomto modulu jsou implementované třídy s metodami pro vytvoření a smazání snímku při použití LVM nebo LINSTOR. Vytvoření snímku je zavoláno vždy před spuštěním procesu zálohy, a smazání snímku po jejím dokončení či selhání.

progress

Implementace jednoduchého textového ukazatele průběhu (angl. progress baru), který je využíván modulem `io`.

qemu_backuper

Jedná se o vstupní bod celé aplikace. Na základě zpracovaných argumentů příkazové řádky modulem `args` jsou volány funkce z ostatních popisovaných modulů.

config

Implementace spravující perzistentní konfiguraci celého nástroje. Zároveň je zde implementována automatická konfiguraci disků, pokud je nástroj využíván v kombinaci s Proxmox VE a LINSTOR.

qemu

Modul `qemu` implementuje jednoduchého klienta pro komunikaci s QEMU pomocí QMP API socketu.

5.4.3 Konfigurace

Konfigurace nástroje probíhá pomocí souboru ve formátu JSON, který obsahuje nastavení jednotlivých virtuálních strojů a jejich disků. Tento formát byl zvolen z důvodu jeho snadné manuální i strojové editace, a možnosti uložení libovolně zanořených datových struktur. Konfigurační soubor obsahuje všechny potřebné pro vytvoření zálohy všech disků virtuálního stroje.

```
{
  "vms": {
    "100": {
      "monitor": "/run/qemu-server/100.qmp",
      "platform": "proxmox",
      "drives": [
        {
          "type": "linstor",
          "identifier": "/dev/drbd/by-res/vm-100-disk-0/0",
          "filename": "/dev/drbd/by-res/vm-100-disk-0/0",
          "dirty_bitmap_path": "/etc/pve/bitmaps/gendjfaj.
              ↪ bitmap",
          "dirty_bitmap_granularity": 65536,
          "linstor_snapshot_name": "qemu_backuper_gavjykvowa"
        }
      ]
    }
  }
}
```

Ukázka kódu 5.2: konfigurace v případě použití s Proxmox VE a LINSTOR

V kořenovém objektu konfigurace se nachází parametr `vms`, který obsahuje seznam všech virtuálních strojů pod jejich vlastním názvem. V případě manuálního používání nástroje z příkazové řádky je možné zvolit libovolný název, jelikož slouží pouze k načtení správné konfigurace pro požadovaný virtuální stroj. Pokud je nástroj využíván spolu s Proxmox VE, tak jsou názvy voleny automaticky podle identifikátoru virtuálního stroje. Pro každý virtuální stroj jsou specifikovány následující parametry:

- **monitor** - Cesta k socketu QEMU monitoru, který umožňuje komunikaci pomocí QMP API za účelem exportu a vymazání dirty bitmapy.
- **platform** - Název platformy, kterou virtuální stroj využívá. V současné chvíli je podporovanou platformou pouze Proxmox VE, avšak v případě používání nástroje přímo s QEMU/KVM bez konkrétní platformy je možné zadat hodnotu `null`.
- **drives** - Pole obsahující konfiguraci všech disků pro virtuální stroj.
  - **type** - Typ disku, podle kterého je následně řízeno vytvoření jeho snímku. Aktuálně jsou podporovány hodnoty `lvm` a `linstor`.
  - **identifier** - Identifikátor disku, který je v QEMU používán k nalezení požadovaného disku. Hodnota může být shodná s `filename` nebo může obsahovat přímo hodnotu `node-name` z konfigurace QEMU, kterou lze manuálně zvolit v případě spouštění QEMU z příkazové řádky.
  - **filename** - Cesta k souboru nebo blokovému zařízení s virtuálním diskem ve formátu RAW.
  - **dirty_bitmap_path** - Cesta k dirty bitmapě pro účely zálohy při vypnutém virtuálním stroji.
  - **dirty_bitmap_granularity** - Granularita dirty bitmapy pro účely inkrementálních záloh.
  - Parametry použité pouze v případě použití se správou logických disků pomocí LVM
    - **lvm_snapshot_size** - Velikost snímku, který bude vytvořen po dobu zálohy. Tato hodnota omezuje kolik dat může virtuální stroj zapsat v jejím průběhu. Doporučená velikost je shodná jako je velikost disku, aby v průběhu zálohy nemohlo dojít k vyčerpání dostupné kapacity, což by vedlo ke znemožnění zápisu dat virtuálním strojem.

    - **via_lvm_snapshot** - Cesta ke snímku, který bude vytvořen před provedením zálohy. V případě použití s Proxmox VE je generována automaticky nahrazením poslední části cesty za náhodný řetězec o délce deseti znaků, čímž je předcházeno vzniku kolizí.
  - Parametry použité pouze v případě použití s distribuovaným síťovým úložištěm LINSTOR
    - **linstor_snapshot_name** - Název snímku, který bude vytvořen před provedením zálohy. V případě použití s Proxmox VE je vygenerován automaticky složením prefixu `qemu_backuper_` a náhodného řetězce o délce deseti znaků.

Konfigurace nástroje je ukládána do souboru `qemu-backuper.json`, a obsahuje konfiguraci všech zálohovaných virtuálních strojů. Pokud je nástroj používán z prostředí příkazové řádky s QEMU/KVM, tak je konfigurační soubor uložen ve složce `/etc`.

V případě používání nástroje skrz administrační rozhraní platformy Proxmox VE je konfigurační soubor umístěn ve složce `/etc/pve`, která je ve výchozím stavu sdílena napříč celým Proxmox VE clusterem. Díky tomu je umožněno využívání živé migrace virtuálního stroje i s vyvinutým řešením bez nutnosti další konfigurace. Požadovaným předpokladem je pouze nasazení vyvinutého řešení na všechny hypervizory, které zároveň musí využívat shodné síťové úložiště pro ukládání záloh. Pokud nový hypervizor využívá jiné úložiště, tak není umožněno navázání na stávající sekvenci inkrementálních záloh, a je nutné provést vytvoření plné zálohy.

Živou migraci virtuálního stroje je možné využívat i se samotným hypervizorem QEMU/KVM, avšak je nutné zajistit sdílení konfigurace vyvinutého nástroje mezi všemi hypervizory v clusteru.

5.4.4 Záloha dat

Základní funkcionalitou nástroje je vytvoření plné a inkrementální zálohy všech disků virtuálního stroje. Pro tento účel je nutné, aby všechny zálohované disky využívaly úložiště LVM nebo LINSTOR, jelikož je pro výrazné zvýšení konzistence zálohy vyžadována možnost vytvoření snímku zálohovaného disku (viz 2.4.2), což vede k zajištění neměnnosti dat v průběhu vytváření zálohy.

I přes to zde existuje potenciální šance zálohy dat v nekonzistentní podobě, kterou je možné dále snížit při záloze s využitím agenta spuštěním

příkazu `sync`, který vynutí zápis všech dat z I/O cache na disk. Avšak vyvinuté řešení agenta nevyužívá, a nemá žádnou možnost jak získat přístup do virtuálního stroje, kvůli čemuž není možné tuto metodu pro zajištění konzistence dat využít. Proto je pro zvýšení konzistence zálohy využíváno dočasného pozastavení virtuálního stroje (viz dále).

Před zmíněným vytvořením snímku dochází v případě tvorby inkrementální zálohy k získání obsahu dirty bitmapy, která je následně vynulována, což umožní zaznamenávání nových zápisů dat pro účely příští inkrementální zálohy. Tyto dvě operace jsou prováděny atomicky, a virtuální stroj v jejich průběhu nemá několik jednotek milisekund přístup k disku. Pokud by tomu tak nebylo, a virtuální stroj mezi nimi provedl zápis na disk, tak by kvůli následnému vynulování dirty bitmapy došlo ke ztrátě informace o tomto zápisu, jehož důsledkem by bylo poškození všech následujících inkrementálních záloh až do následující plné zálohy.

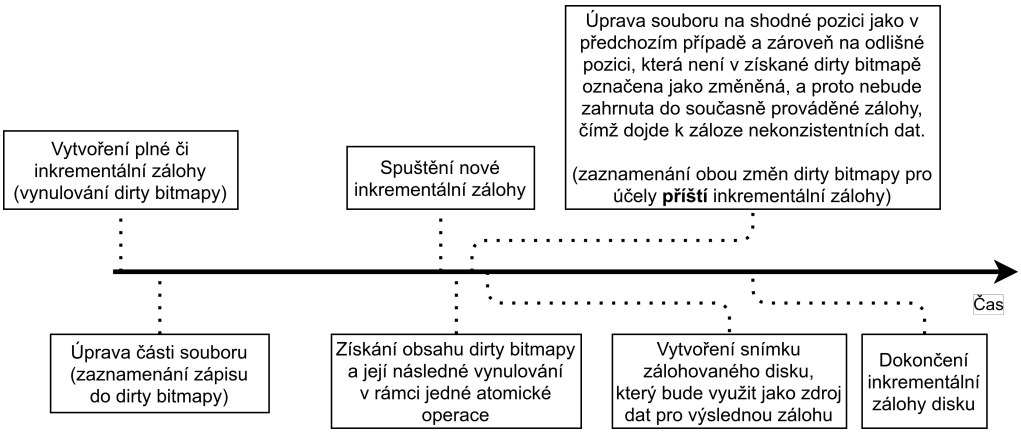
Do shodné atomické operace je zahrnuto i vytvoření snímku všech zálohovaných disků, čehož je docíleno pozastavením virtuálního stroje před získáním obsahu dirty bitmapy, a následným spuštěním po vytvoření snímků všech disků. Pozastavení virtuálního stroje sice vede k nedostupnosti virtuálního stroje v řádu sekund, avšak předchází poslední z následujících situací, které by mohly nastat mezi získáním obsahu dirty bitmapy a vytvořením snímku zálohovaného disku:

- Mezi získáním obsahu dirty bitmapy a vytvořením snímku neproběhne zápis dat na disk, díky čemuž nedojde k ovlivnění výsledné zálohy.
- Mezi získáním obsahu dirty bitmapy a vytvořením snímku proběhne zápis dat na disk, ale na pozici, která nebude zahrnuta do inkrementální zálohy. Výsledná záloha opět není nijak ovlivněna.
- Mezi získáním obsahu dirty bitmapy a vytvořením snímku proběhne zápis dat na pozici disku, která bude zahrnuta do inkrementální zálohy. V tomto případě sice dojde k ovlivnění obsahu zálohy, která bude obsahovat o pár sekund novější data, avšak stále nedojde k ovlivnění konzistence, jelikož dojde k záloze všech zapsaných dat.
- Potenciálním problémem je poslední možná situace, kdy je mezi získáním obsahu dirty bitmapy a vytvořením snímku proveden zápis dat na dvě různé pozice disku, z nichž bude na základě získané dirty bitmapy do inkrementální zálohy zahrnuta pouze jedna. V tomto případě dojde k záloze dat v nekonzistentním stavu.

  Pokud uvážíme, že obě zápisové operace mění jeden soubor na zálohovaném disku, čímž z principu vzniká jejich vzájemná závislost,

přičemž první zápisová operace změní data, která nebyla v čase od poslední zálohy do získání obsahu dirty bitmapy změněna, a druhá operace provede změnu dat, která od poslední zálohy změněna byla, tak je do nové zálohy zahrnuta pouze změna z druhé zápisové operace. Výsledkem je záloha nové podoba souboru, která obsahuje kombinaci nových a starých dat.

Pro lepší nastínění problému je popsaná situace vizualizována pomocí časové osy na obr. 5.3.



Obrázek 5.3: Vizualizace vzniku nekonzistentní zálohy bez získání obsahu dirty bitmapy a vytvoření snímku disku v rámci jedné atomické operace

Umístění výsledné zálohy je určeno parametrem příkazové řádky `--backup-path-prefix`, ve kterém je vytvořena složka s názvem virtuálního stroje, která dále obsahuje složku s názvem zálohovaného disku, ve které jsou uloženy samotné zálohy disku.

Vytvoření inkrementální zálohy předem nakonfigurovaného (viz 5.4.3) virtuálního stroje s názvem `debian10` je možné spustit z příkazové řádky pomocí následujícího příkazu:

```
qemu-backuper debian10 inc --backup-path-prefix /mnt/backups
```

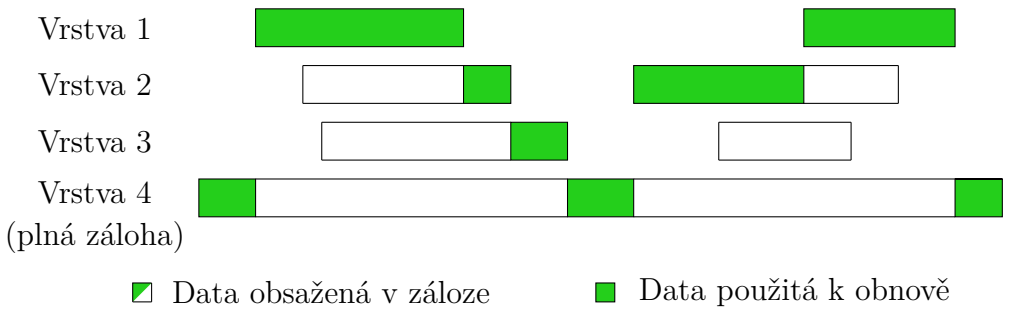
Ukázka kódu 5.3: Vytvoření inkrementální zálohy virtuálního stroje

V případě požadavku na vytvoření plné zálohy je nutné parametr `inc` nahradit parametrem `full`. Kompletní dokumentace dostupných parametrů zálohy je obsažena v příloze C.

5.4.5 Obnova dat

Další funkcionalitou nástroje je obnova disku z plné či inkrementální zálohy, která může být provedena do původního nebo nového úložiště.

Obnova dat funguje na principu sestavení všech záloh (od poslední plné až do vybrané) do objektu reprezentujícího virtuální disk, kde jedna záloha reprezentuje jednu vrstvu disku, přičemž novější vrstva má vyšší prioritu než starší. Při požadavku na čtení z tohoto objektu je na základě priorit vybrána vrstva, ze které budou data přečtena (viz 5.4). Toto řešení umožňuje rychlou kombinaci více záloh bez nutnosti jejich předchozího uložení do nového souboru.



Obrázek 5.4: Příklad virtuálního disku vytvořeného k obnově

Tato funkcionalita je zamýšlena především pro obnovu původního disku, kterou je možné provést pouze v případě vypnutého virtuálního stroje. Pro účely procházení obsahu zálohy je ale možné sestavit výslednou podobu zálohy do libovolného umístění, ze kterého může být následně připojena pomocí běžných nástrojů operačního systému GNU/Linux.

Obnovení zálohy do původního úložiště je možné spustit následujícím příkazem:

```
qemu-backuper debian10 build-img --restore
    --backup-path-prefix /mnt/backups
    --drive /dev/drbd/by-res/vm-100-disk-0/0
    --datetime 2021-05-04T14:12:00
    --output /dev/drbd/by-res/vm-100-disk-0/0
```

Ukázka kódu 5.4: Obnova zálohy do původního virtuálního disku

- `debian10` - Název virtuálního stroje, pro který byla vytvořena záloha
- `build-img` - Požadavek na sestavení obrazu disku ze zálohy
- `--restore` - Předání informace o tom, že se jedná o obnovu dat na původní disk. Na základě tohoto parametru bude příště vynuceno provedení plné zálohy, jelikož po obnově není možné jednoduše navázat na předchozí sekvenci inkrementálních záloh.

- `--backup-path-prefix` - Cesta ke kořenové složce obsahující zálohy virtuálních strojů
- `--drive` - Absolutní cesta k disku, který bude obnovován (shodná jako parametr `filename` v konfiguračním souboru)
- `--datetime` - Čas ve formátu ISO 8601 s přesností na sekundy, na základě kterého bude vybrána použitá záloha pro sestavení obrazu disku. Pokud neexistuje záloha ve zvoleném čase, tak je vybrána nejbližší starší.
- `--output` - Umístění výsledného obrazu disku, který je možné uložit do blokového zařízení nebo samostatného souboru. Při obnově zálohy na původní disk by mělo být shodné jako hodnota parametru `--drive`.

5.5 Integrace do Proxmox Virtual Environment

Poslední fází vývoje byla integrace celého řešení do virtualizační platformy Proxmox VE ve verzi 6.3-1. Ačkoliv je velmi pravděpodobné, že vyvinutý patch půjde aplikovat i na následující verze Proxmoxu VE, tak není možné tuto kompatibilitu zaručit, a proto je pro bezproblémové nasazení doporučena zmíněná verze.

Integrace se skládala především z rozšíření platformy o podporu pro vyvinutý nástroj, a integrace přímo do webové administrace. Tato modifikace je instalována metodou aplikace patche pomocí nástroje `patch`, díky čemuž nebylo v rámci práce potřeba modifikovat instalační obraz Proxmoxu VE. Tento patch je dodáván spolu s textem bakalářské práce na přiloženém CD v souboru `pve.patch` (viz 6.1.3).

V rámci aplikační logiky napsané v programovacím jazyce Perl 5 byl upraven zdrojový kód sloužící k vytváření a obnově záloh. K tomuto účelu byl využit již stávající modul `VZDump`, který implementuje podporu pro zálohování stejnojmenným nástrojem. V tomto modulu byly upraveny soubory `VZDump/Common.pm` a `VZDump/QemuServer.pm`.

V prvním souboru byla implementována podpora pro plánování vytvoření záloh vybraných virtuálních strojů ve zvoleném čase, což zahrnovalo především vygenerování příkazu pro spuštění vyvinutého nástroje místo výchozího nástroje `VZDump`.

V druhém souboru `VZDump/QemuServer.pm` byla rozšířena pouze funkce `restore_vma`, která slouží k provedení zálohy zvoleného virtuálního stroje.

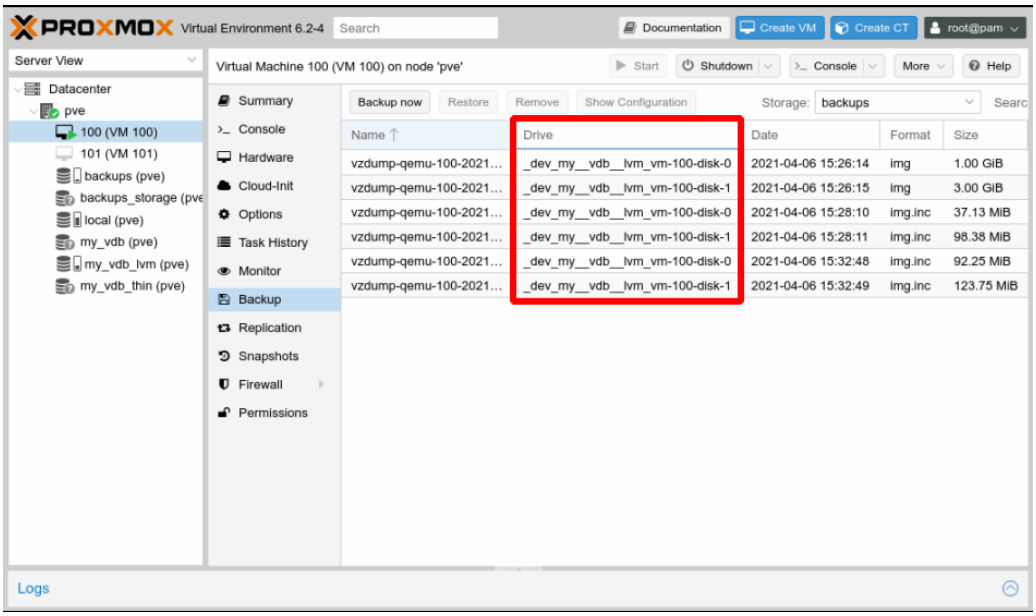
V rámci této funkce bylo přidáno podmíněné spouštění vyvinutého nástroje, který je využit v případě, že je pro vybraný virtuální stroj nakonfigurován.

V rámci integrace úprav do rozhraní webové administrace byly dále upraveny moduly `QemuServer`, `Storage` a `API2`.

V prvním modulu došlo k úpravě souborů `QemuServer/Drive.pm` a `QemuServer.pm`. Úpravy v prvním souboru přidávají možnost konfigurace nově přidaných parametrů disku (obr. 6.1 a 6.2). V souboru `QemuServer.pm` byla implementována podpora pro uložení nových parametrů do konfiguračního souboru virtuálního stroje, a jejich opětovné načtení. Dále je v tomto souboru implementována podpora pro automatickou konfiguraci disků ve vyvinutém zálohovacím nástroji, a také spouštění samotné obnovy vybrané zálohy do původního disku.

V modulu `Storage` byly upraveny soubory `Storage.pm` a `Storage/Plugin.pm`. V prvním souboru došlo k rozšíření regulárního výrazu pro vyhledávání souborů se zálohami, do kterého byla přidána podpora pro novou příponu záloh `.img.inc`, která je používána pro inkrementální zálohy. Ve druhém souboru byla implementována podpora pro adresářovou strukturu umístění záloh, která je vyvinutým nástrojem členěna do složek na základě názvu virtuálního stroje a absolutní cesty k zálohovanému disku. Dále byla v tomto souboru rozšířena funkce pro mazání záloh, která v případě smazání jedné zálohy smaže všechny i zálohy, které jsou na smazané záloze závislé.

V posledním zmíněném modulu `API2` byly upraveny soubory `API2/Qemu.pm` a `API2/Storage/Content.pm`. V souboru `API2/Qemu.pm` byla přidána podpora pro předání názvu obnovovaného disku z webového rozhraní, který je následně využit v souboru `QemuServer.pm` pro spuštění obnovy disku. Druhý soubor `API2/Storage/Content.pm` implementuje změny související s mazáním zálohy z webové administrace. Tyto změny zahrnují předávání identifikátoru virtuálního stroje do funkce `vdisk_free` v souboru `Storage/Plugin.pm`, jelikož vyvinutý nástroj vyžaduje tuto informaci ke správnému načtení konfigurace pro konkrétní virtuální stroj.



Obrázek 5.5: Upravený seznam provedených záloh

Samotné úpravy webové administrace byly provedeny v souboru `js/pvemanagerlib.js`. Předmětem úprav bylo především přidání nových polí do zmíněného průvodce pro vytvoření virtuálního stroje. Další vizuální úprava spočívala v upravení seznamu provedených záloh, kam byl přidán sloupec `Drive` (obr. 5.5), který zobrazuje pro jaký disk byla záloha vytvořena, díky čemuž je umožněna obnova pouze zvoleného disku.

6 Instalace a ověření funkčnosti řešení

Pro ověření funkčnosti vyvinutého řešení je potřeba provést nejprve jeho instalaci, která se skládá z více částí. První je instalace upravené verze QEMU, která implementuje rozšířené možnosti práce s dirty bitmapami (viz 5.3). Následně je nutné provést instalaci vyvinutého nástroje, který slouží k vytváření a obnovu plných a inkrementálních záloh virtuálních strojů v QEMU/KVM (viz 5.4). Poslední částí je aplikování patche na existující instalaci Proxmoxu VE, který přidává podporu pro zálohování virtuálních strojů provozovaných na této platformě přímo z jejího webového rozhraní.

6.1 Instalace

Instalace vypracovaného řešení se skládá z instalace upravené verze QEMU do prostředí Proxmox Virtual Environment, instalace zálohovacího nástroje, a aplikování patche na Proxmox VE. Všechny uvedené příkazy jsou prováděny z domovského adresáře uživatele `root` (`/root`), pokud není uvedeno jinak. V příkladech je jako IP adresa vzdáleného Proxmox VE serveru uvedena `192.168.1.131`, kterou je potřeba změnit na základě vlastní konfigurace sítě. Cesta `/media/cdrom` označuje kořenový adresář přiloženého CD. Dále uvedený postup předpokládá, že je instalace prováděna z operačního systému GNU/Linux, na kterém jsou nainstalovány nástroje `scp` a `ssh`.

6.1.1 Instalace QEMU v prostředí Proxmox VE

Instalace upravené verze QEMU probíhá z připraveného instalační souboru `pve-qemu-kvm_5.2.0-5_amd64.deb` na přiloženém CD, který je nejdříve potřeba zkopírovat na Proxmox VE server pomocí nástroje `scp`.

```
user@local:/media/cdrom# scp pve-qemu-kvm_5.2.0-5_amd64.deb
    root@192.168.1.131:/root
pve-qemu-kvm_5.2.0-5_amd64.deb     100% 21MB    2.9MB/s    00:09
```

Ukázka kódu 6.1: Ukázka zkopírování instalačního souboru QEMU na server

Nyní se připojíme na server, a nainstalujeme zkopírovaný instalační soubor pomocí nástroje `dpkg`. Poté je nutné restartovat službu `pvedaemon` pomocí nástroje `systemctl`, čímž dojde k použití upravené verze QEMU. Po-

kud již v této fázi byly spuštěny nějaké virtuální stroje, tak je potřeba provést jejich vypnutí a opětovné zapnutí, bez kterého pro tyto virtuální stroje nedojde k aplikování změn.

```
user@local:/media/cdrom# ssh root@192.168.1.131 # Připojení na server
root@pve:~# dpkg -i pve-qemu-kvm_5.2.0-5_amd64.deb
(Reading database ... 121632 files and directories currently
    ↪ installed.)
Preparing to unpack pve-qemu-kvm_5.2.0-5_amd64.deb ...
Unpacking pve-qemu-kvm (5.2.0-5) over (5.2.0-5-7) ...
Setting up pve-qemu-kvm (5.2.0-5) ...
... (zkráceno)
root@pve:~# systemctl restart pvedaemon pveproxy
root@pve:~#
```

Ukázka kódu 6.2: Ukázka instalace upravené verze QEMU

V aktuální fázi je nainstalována upravená verze QEMU implementující potřebné změny. Úspěšnost instalace můžeme ověřit tak, že za pomoci webové administrace Proxmox VE vytvoříme a spustíme libovolný virtuální stroj s pevným diskem, na který není potřeba instalovat operační systém. Následně je možné na server zkopírovat připravený skript pro ověření korektní instalace QEMU, který je na přiloženém CD uložen pod názvem `check-qemu-install.sh`. Tento skript následně spustíme, a jako parametr předáme cestu ke QMP socketu, který je v případě platformy Proxmox VE umístěn v adresáři `/run/qemu-server`, a jeho název je složen z identifikátoru virtuálního stroje a přípony `qmp`.

```
user@local:/media/cdrom# scp -r check-qemu-install.sh
    root@192.168.1.131:/root
user@local:/media/cdrom# ssh root@192.168.1.131 # Připojení na server
root@pve:~# bash ./check-qemu-install.sh /run/qemu-server/100.qmp
Modified version of QEMU has been installed successfully!
```

Ukázka kódu 6.3: Ověření správné kompilace a instalace `pve-qemu`

Pokud bude výstup posledního příkazu shodný s uvedeným příkladem, tak byla upravená verze QEMU nainstalována úspěšně. V opačném případě bude vypsán text *ERROR*, který signalizuje, že byla instalace provedena chybně, a je nutné ji opakovat.

6.1.2 Instalace nástroje pro vytváření záloh

Na stroj s nainstalovaným Proxmox VE zkopírujeme adresář `qemu-backuper`, připojíme se na server, a nainstalujeme balíček balíček `python3-setuptools`, který umožní instalaci vyvinutého nástroje. Následně se pomocí nástroje `cd`

přesuneme do složky `qemu-backuper`, a spustíme instalaci nástroje pomocí příkazu `python3 setup.py install -f`.

Úspěšnost instalace lze ověřit spuštěním příkazu `qemu-backuper --help`, který vypíše nápovědu k tomuto nástroji.

```
user@local:/media/cdrom# scp -r qemu-backuper root@192.168.1.131:/root
config.py                          100 8386 7.2MB/s 00:00bitmap.py
   100 3092     4.6MB/s   00:00
... (zkráceno)
user@local:/media/cdrom# ssh root@192.168.1.131 # Připojení na server
root@pve:~# apt install python3-setuptools
Reading package lists... Done
Building dependency tree
... (zkráceno)
Unpacking python3-setuptools (40.8.0-1) ...
Setting up python3-setuptools (40.8.0-1) ...
root@pve:~# cd qemu-backuper
root@pve:~/qemu-backuper# python3 setup.py install -f
running install
running bdist_egg
... (zkráceno)
Processing dependencies for qemu-backuper==1.0.0
Finished processing dependencies for qemu-backuper==1.0.0
root@pve:~/qemu-backuper# qemu-backuper --help
usage: qemu-backuper [-h] [-p [{proxmox}]]
            [--config [CONFIG]]
            vm_name {full,inc,config,build-img,check} ...
positional arguments:
vm_name Name of the virtual machine. In proxmox it's an VM ID
... (zkráceno)
```

Ukázka kódu 6.4: Instalace nástroje pro vytváření záloh

6.1.3 Integrace řešení do Proxmox VE

Nyní je možné provést aplikaci patche na Proxmox VE, díky které bude možné spravovat zálohy přímo z webové administrace. Proxmox VE musí být nainstalován pomocí oficiálního instalačního obrazu ve verzi 6.3-1 [91] bez provedených aktualizací. Pokud tato podmínka nebude dodržena, tak hrozí riziko nemožnosti aplikování patche.

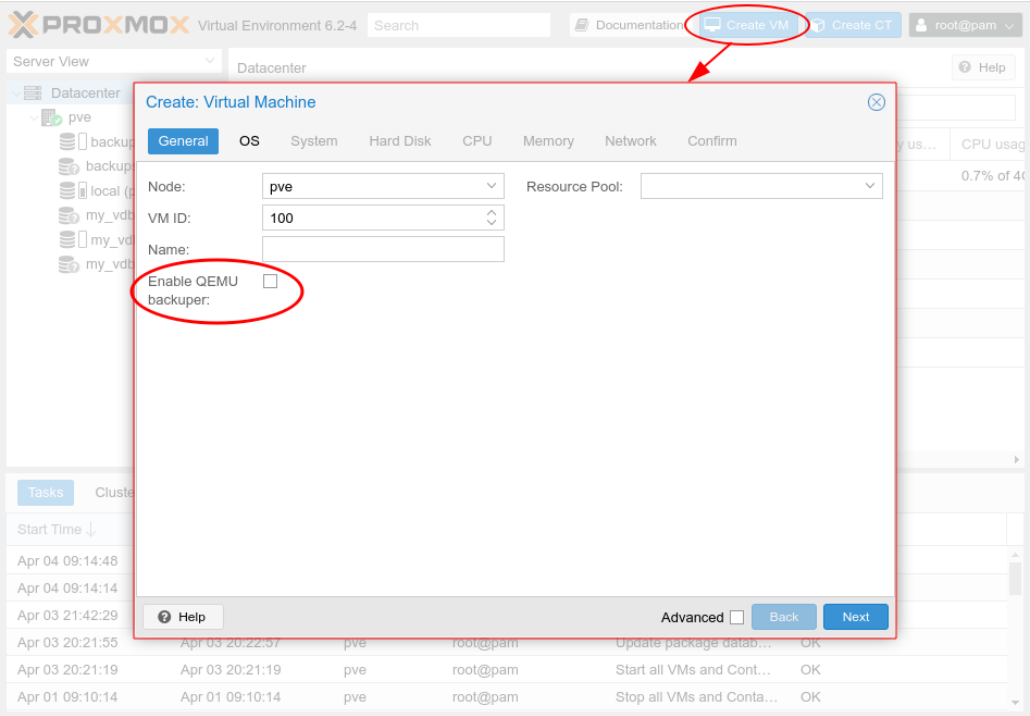
Na stroj, kde je nainstalován Proxmox VE zkopírujeme patch `pve.patch` z přiloženého CD, a následně ho aplikujeme pomocí nástroje `patch`. Poté je opět vyžadováno restartování služeb `pvedaemon` a `pveproxy`.

```
user@local:/media/cdrom# scp -r pve.patch root@192.168.1.131:/root
```

```
pve.patch                          100%   18KB 965.0KB/s   00:00
root@pve:~# patch --forward -p1 --batch -d /usr/share/ < pve.patch
patching file usr/share/perl5/PVE/API2/Qemu.pm
... (zkráceno)
patching file usr/share/perl5/PVE/VZDump.pm
patching file usr/share/pve-manager/js/pvemanagerlib.js
root@pve:~# systemctl restart pvedaemon pveproxy
root@pve:~#
```

Ukázka kódu 6.5: Ukázka úspěšné aplikace patche

Pokud byla aplikace patche úspěšná, tak se po kliknutí na tlačítko `Create VM` v pravém horním rohu zobrazí modifikovaný formulář pro vytvoření virtuálního stroje (obr. 6.1).



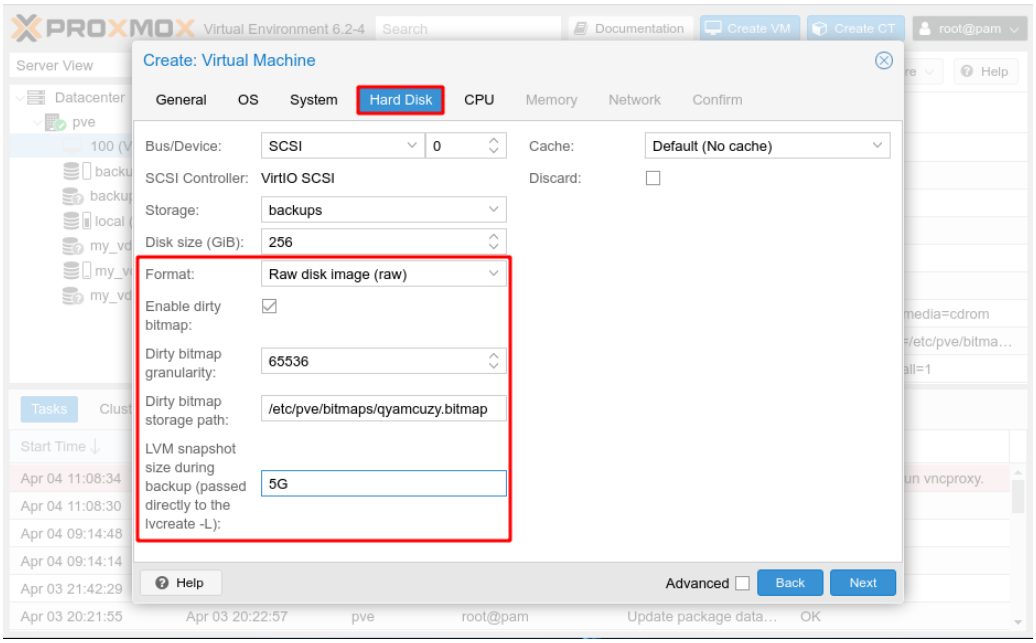
Obrázek 6.1: Modifikovaný formulář pro vytvoření virtuálního stroje na platformě Proxmox VE

6.2 Nasazení do prostředí virtuálního stroje

Nasazení vyvinutého řešení pro virtuální stroj je plně integrováno do webové administrace Proxmox VE, a jeho konfigurace je součástí formuláře pro vytvoření nového virtuálního stroje.

6.2.1 Konfigurace virtuálního stroje v Proxmox VE

Aby bylo možné zálohovat virtuální stroj pomocí vyvinutého řešení, tak je na úvodní obrazovce vytváření virtuálního stroje (obr. 6.1) nutné zaškrtnout pole `Enable QEMU backuper`. Dále je potřeba na záložce `Hard Disk` vybrat formát RAW (viz 5.1), a zaškrtnout pole `Enable dirty bitmap`. V následujících textových polích je možné konfigurovat jednotlivé parametry dirty bitmapy a LVM snímku (obr. 6.2). Na závěr je nutné provést spuštění nově vytvořeného virtuálního stroje, čímž dojde k automatické konfiguraci nástroje pro vytváření záloh. Bez tohoto kroku nebude záloha pomocí vyvinutého nástroje provedena (viz. 6.2.2).



Obrázek 6.2: Modifikovaný formulář pro definici disku při vytváření virtuálního stroje v platformě Proxmox VE

Pokud nebylo zálohování pomocí vyvinutého řešení zapnuto při vytváření virtuálního stroje, tak je stále možné provést dodatečnou konfiguraci skrz odpojení disku pomocí tlačítka `Detach`, čímž vznikne `Unused Disk`. Následně se po kliknutí na tlačítko `Edit` zobrazí formulář pro opětovné připojení disku k virtuálnímu stroji, kde je mj. umožněna kompletní konfigurace dirty bitmapy podobně jako je tomu u přidávání úplně nového disku.

Pomocí formulářů pro vytvoření nového virtuálního stroje i přidání nového disku do již existujícího virtuálního stroje je možné konfigurovat granularitu dirty bitmapy, cestu na souborovém systému, kde bude uložena, a pokud je pro virtuální disk využito úložiště LVM, tak je dále možné nastavit

velikost snímku, který bude vytvořen před spuštěním procesu zálohy.

Dirty bitmap granularity

Parametr definující granularitu dirty bitmapy, která je ve výchozím stavu nastavena na doporučenou hodnotu 65536 (viz 2.2.1).

Dirty bitmap storage path

Cesta pro persistentní uložení dirty bitmapy při vypnutí virtuálního stroje. Ve výchozím stavu je nastavena do složky `/etc/pve/bitmaps`, která je sdílena napříč celým Proxmox VE clusterem, což umožní snadnou migraci virtuálního stroje na jiný hypervizor v clusteru. Název souboru je ve výchozím stavu náhodně vygenerován, a jeho manuální zvolení není vyžadováno.

LVM snapshot size during backup

Tento parametr definuje velikost LVM snímku, a omezuje množství dat, které může virtuální stroj zapsat v průběhu zálohy. Hodnota je přímo předávána pomocí parametru `-L` příkazu `lvcreate`, a je nutné za číselnou hodnotu doplnit i její jednotku [92]. Pokud je detekováno použití úložiště LINSTOR, tak je toto konfigurační pole skryto.

6.2.2 Konfigurace nástroje pro vytváření záloh

Nástroj pro správu záloh je v případě použití s platformou Proxmox VE nakonfigurován plně automaticky při každém spuštění virtuálního stroje, a není tudíž potřeba žádná manuální konfigurace.

Tato autokonfigurace zajistí detekci cesty k socketu QMP monitoru, a následné přidání všech disků včetně jejich konfigurace. V případě, že autokonfigurace selže, tak je proces spouštění virtuálního stroje přerušen, a uživatel je o této skutečnosti informován červeným záznamem ve spodní části webové administrace v sekci `Logs - Tasks`, kde je možné dvojitým kliknutím zobrazit podrobné informace o chybě. Díky tomu je zajištěna spolehlivá funkčnost celého řešení, a korektní konfigurace každého spuštěného virtuálního stroje.

7 Výkonnostní testy

Vyvinuté řešení bude v rámci výkonnostních testů srovnáno s výchozím nástrojem VZDump, který je možné v Proxmox VE využívat bez nutnosti instalace dalších nástrojů, a nově vyvinutým oficiálním řešením Proxmox Backup Server (viz 4.2.2), který již na rozdíl od VZDumpu podporuje vytváření inkrementálních záloh.

Dále bude do srovnání zahrnut komunitní patch pro VZDump [93], který přidává podporu pro vytváření diferenciálních záloh, a funguje na principu srovnávání zálohovaného disku s poslední plnou zálohou pomocí nástroje `pve-xdelta3` [94]. Ačkoliv je tato metoda tvorby diferenciální zálohy výpočetně náročná, tak se do vydání Proxmox Backup Serveru jednalo o jediný způsob, kterým bylo v Proxmox VE možné vytvářet jiný než plný typ záloh.

7.1 Specifikace prostředí

Veškerá měření budou probíhat na serveru DELL PowerEdge R820 se čtyřmi fyzickými procesory Intel Xeon E5-4620 v2 na frekvenci 2.6 GHz a 251 GiB RAM typu DDR3. Pro disky virtuálních strojů je využito distribuované síťové disku LINSTOR, které je nainstalováno na shodném serveru, a využívá hardwarový RAID řadič DELL PERC H710 s dostupnou kapacitou 557 GiB, ze kterého je alokováno 500 GiB pouze pro LINSTOR. Pro ukládání záloh je využito síťové úložiště NFS o celkové kapacitě 1 TiB, které je k zálohovanému serveru připojeno síťovou linkou o rychlosti 10 Gb/s, a dosahuje zápisu o rychlosti 109 MB/s a čtení o rychlosti 117 MB/s.

7.2 Metodika měření

Předmětem zálohy je vždy nesystémový disk spuštěného virtuálního stroje v Proxmox VE, díky čemuž je možné přesně kontrolovat zápis dat, jelikož nainstalovaný operační systém neovlivňuje měření svými vlastními zápisy. Použitý virtuální stroj má k dispozici virtuální osmijádrový procesor a 8 GB RAM. Před každou zálohou je disk zaplněn náhodnými daty z blokového zařízení `/dev/urandom` pomocí následujícího příkazu:

```
dd if=/dev/urandom of=/dev/vdb bs=4M
```

Ukázka kódu 7.1: Příkaz použitý pro zaplnění celého disku náhodnými daty

Pokud je disk zaplněn jen z části, tak je proveden opakovaný zápis náhodných dat o velikosti 4 MiB na náhodné pozice disku. V použitém skriptu je parametrem `WRITE_COUNT` nastaven počet opakování náhodných zápisů, který je vypočten jako podíl požadovaného množství zapsaných dat a velikostí zapsaných dat v jednom cyklu. Pro zápis 1 GB dat je hodnota stanovena na 250, pro zápis 3 GB dat na 750, a pro zápis 30 GB dat na 7500.

```
# Získání celkové velikosti zálohovaného disku
DISK_SIZE_BYTES=$(($(cat /sys/class/block/vdb/size)*\
    $(cat /sys/class/block/vdb/queue/physical_block_size)))

# Počet zápisových operací o~velikosti 4 MiB (zápis 1 GB dat)
WRITE_COUNT=250

# Velikost zapsaných dat v~jednom cyklu
BLOCK_SIZE=4194304 # 4 MiB

# Cyklus pro zápis požadovaného počtu bloků na náhodné pozice
   ↪ disku
for RANDOM_POS in \
    `shuf -i 0-$(($DISK_SIZE_BYTES/$BLOCK_SIZE)) \
        -n $WRITE_COUNT`; do

    dd if=/dev/urandom of=/dev/vdb \
        bs=$BLOCK_SIZE count=1 \
        oflag=seek_bytes seek=$(($RANDOM_POS*$BLOCK_SIZE))
done
```

Ukázka kódu 7.2: Skript použitý pro zaplnění části disku náhodnými daty

Pokud je v rámci měřeného scénáře sledován i čas zálohy při měření poklesu dostupného výkonu čtení a zápisu ve virtuálním stoji, tak jsou tyto testy provedeny v samostatných opakováních.

V rámci prvního testu je vždy sledován pouze celkový čas zálohy disku bez dalších vlivů. V druhém testu je po spuštění zálohy spuštěn test výkonu zápisu pomocí nástroje `dd`. Účelem je zkoumání poklesu výkonu zápisu ve virtuálním stroji v průběhu zálohy oproti naměřené referenční hodnotě (viz dále), a vliv tohoto výkonnostního testu na celkový čas zálohy. V posledním opakování je v průběhu zálohy spuštěno měření výkonu čtení pomocí nástroje `hdparm`, jehož účelem je zjištění poklesu výkonu čtení ve virtuálním stroji vůči naměřenému referenčnímu výkonu (viz dále), a jeho vliv na čas zálohy. Jelikož test `hdparm` trvá pouze jednotky sekund, tak je spouštěn v cyklu po celou dobu zálohy, což umožňuje pozorovat vliv tohoto testu na čas zálohy. Uvedené hodnoty naměřeného dostupného výkonu čtení jsou vypočteny jako aritmeticky průměr ze všech cyklů.

```
# Měření výkonu zápisu:
dd if=/dev/zero of=/dev/vdb bs=4M status=progress

# Měření výkonu čtení:
hdparm -Tt /dev/vdb
```

Ukázka kódu 7.3: Příkazy použité pro měření výkonu zápisu a čtení

U všech zálohovacích řešení kromě Proxmox Backup Serveru, který má vynucený kompresní algoritmus zstd, je komprese záloh vypnuta. Důvodem je, že vyvinuté řešení kompresi dat nepodporuje, a vypnutí komprese u ostatních řešení má za účel zajištění co možná nejpodobnějších podmínek pro všechna zálohovací řešení. Navíc jsou vždy předmětem zálohy náhodná data, u kterých nemá komprese výrazný dopad na výslednou velikost dat, a způsobovala by akorát výkonnostní handicap pro dané řešení.

Aby bylo možné komunitní patch pro VZDump, který podporuje pouze diferenciální zálohy, srovnávat s ostatními řešeními, které podporují pouze inkrementální zálohy, tak je tento typ záloh následuje vždy po vytvoření plné zálohy. Díky tomu jsou oba typy záloh zaměnitelné, a je možné je spolu srovnávat.

U obnovy disku byl vždy měřen pouze celkový potřebný čas bez dalších opakování, a probíhá za vypnutého stavu virtuálního stroje. Před obnovou dat je vždy vypočítán kontrolní součet pro celý disk, který je následně přepsán náhodnými daty. Opětovným vypočtením kontrolního součtu po obnově dat je verifikováno korektní provedení testu.

7.3 Měřené scénáře

Nejprve byl ve virtuálním stroji s diskem ve formátu RAW o kapacitě 100 GB změřen výkon čtení a zápisu, přičemž je výsledek tohoto měření použit jako referenční výkon použitého disku, a slouží k vypočtení procentuálního výkonnostního propadu úložiště ve virtuálním stroji v následujících testech. Výkon zápisu byl měřen v pěti opakováních, přičemž naměřené extrémy byly 319 MB/s a 338 MB/s, a referenční výkon zápisu byl aritmetickým průměrem všech opakování stanoven na 329 MB/s. Výkon čtení byl vzhledem ke krátkému trvání testu měřen ve dvaceti opakováních. Naměřené extrémy dosahovaly hodnot 310 MB/s a 335 MB/s, a aritmetickým průměrem byl referenční výkon čtení stanoven na 321 MB/s.

Veškeré měřené scénáře jsou prováděny pro virtuální disky o kapacitách 10 a 100 GB, které byly zvoleny jako typické velikosti disků v podnikovém prostředí. Disk o velikosti 10 GB reprezentuje např. systémový disk, který je

využíván pouze pro operační systém a nainstalované aplikace. Naopak disk o velikosti 100 GB reprezentuje datové úložiště, které je využíváno např. pro ukládání uživatelských dat, která jsou na server nahrávána koncovými uživateli skrz provozovanou službu.

7.3.1 Plná záloha

Základním scénářem je vytvoření plné zálohy, která je vyžadována pro všechny ostatní typy záloh.

Zálohovací řešení	Čas zálohy bez měření výkonu	Čas zálohy při měření zápisu	Naměřený výkon zápisu ve VM (změna vůči referenčním hodnotám)	Čas zálohy při měření čtení	Naměřený výkon čtení ve VM (změna vůči referenčním hodnotám)
VZDump	1m 51s	2m 7s	87 MB/s (-73,6 %)	2m 11s	243 MB/s (-24,3 %)
Proxmox Backup Server	1m 38s	1m 44s	105 MB/s (-68,1 %)	1m 58s	223 MB/s (-30,5 %)
Komunitní patch pro VZDump	1m 53s	2m 3s	111 MB/s (-66,3 %)	2m 12s	226 MB/s (-29,6 %)
Vyvinuté řešení	1m 51s	2m 44s	120 MB/s (-63,5 %)	1m 50s	180 MB/s (-43,9 %)

Tabulka 7.1: Výsledek měření zálohovacích řešení pro plnou zálohu disku o velikosti 10 GB

Z tabulky naměřených hodnot 7.1 vyplývá, že je vyvinuté řešení při záloze disku o kapacitě 10 GB shodně výkonné jako nástroj VZDump i jeho komunitní patch. Přesto byl nejrychlejší Proxmox Backup Server, který vytvořil plnou zálohu disku o 14 sekund rychleji. Horší výkon u vyvinutého nástroje je způsoben tím, že je před zálohou každého disku vytvořen jeho snímek, což v závislosti na velikosti zálohovaného disku může trvat jednotky až nižší desítky sekund. Konkrétně v tomto scénáři byla naměřena režie na vytvoření snímku 8 sekund. Dále je nutné po dokončení vytváření zálohy tento snímek opět smazat, aby zbytečně nevyužíval fyzické úložiště, což typicky trvá 2-3 sekundy. Tato režie je konstantní bez ohledu na prováděný typ

zálohy, a je tudíž pozorovatelná pouze při takto krátkých časech vytváření zálohy.

Zálohovací řešení	Čas zálohy bez měření výkonu	Čas zálohy při měření zápisu	Naměřený výkon zápisu ve VM (změna vůči referenčním hodnotám)	Čas zálohy při měření čtení	Naměřený výkon čtení ve VM (změna vůči referenčním hodnotám)
VZDump	19m 3s	22m 0s	102 MB/s (-69,0 %)	20m 32s	219 MB/s (-31,8 %)
Proxmox Backup Server	21m 24s	23m 46s	76 MB/s (-76,9 %)	23m 15s	209 MB/s (-34,9 %)
Komunitní patch pro VZDump	19m 9s	22m 3s	104 MB/s (-68,4 %)	20m 28s	208 MB/s (-35,2 %)
Vyvinuté řešení	15m 37s	19m 15s	103 MB/s (-68,7 %)	15m 48s	160 MB/s (-50,2 %)

Tabulka 7.2: Výsledek měření zálohovacích řešení pro plnou zálohu disku o velikosti 100 GB

Plnou zálohu 100 GB disku provedlo vyvinuté řešení nejrychleji, čímž je potvrzeno předchozí tvrzení. Režie na vytvoření a následné smazání snímku je u časů v řádu desítek minut zanedbatelná, díky čemuž se plně projevil výkon vytváření plné zálohy, který je ze všech měřených řešení nejlepší.

Nevýhodou vyvinutého řešení je, že dostupný výkon čtení ve virtuálním stroji omezuje nejvíce, což je dáno především odlišnými technikami tvorby zálohy. Proxmox Backup Server, VZDump i jeho komunitní patch využívají k tvorbě zálohy přímou spolupráci s QEMU. Za předpokladu, že virtuální stroj přečte data z disku, tak jsou tato data využita i k tvorbě zálohy, a tudíž není nutné, aby zálohovací řešení provedlo opakovaný požadavek na přečtení těchto dat z disku. Tuto optimalizaci vyvinuté řešení nevyužívá z důvodu jeho vysoké technické náročnosti, jehož implementace by vyžadovala rozsáhlé úpravy v QEMU a zároveň podporu pro tvorbu zálohy z dat čtených z náhodných pozic disku, která je komplikovaná zejména při tvorbě inkrementální zálohy, kde není předem známa pozice zálohovaných dat v souboru s vytvářenou zálohou.

7.3.2 Vytvoření inkrementální/diferenciální zálohy (1 GB změna dat)

Druhým scénářem je měření inkrementální a diferenciální zálohy při změnách o rozsahu 1 GB dat od poslední plné zálohy. Taková změna dat může odpovídat např. přírůstku souborů s logy za jeden den.

Zálohovací řešení	Čas
VZDump	nepodporuje
Proxmox Backup Server	7s
Komunitní patch pro VZDump	9m 54s
Vyvinuté řešení	15s

Tabulka 7.3: Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 10 GB se změnou 1 GB

Inkrementální zálohu disku o kapacitě 10 GB se změnou 1 GB dat od poslední plné zálohy nejrychleji vytvořil Proxmox Backup Server. Vyvinuté řešení vytvořilo zálohu za dvojnásobný čas, což je opět dáno režií pro vytvoření a smazání snímku zálohovaného disku, která se nejvíce projevuje při takto krátkých časech zálohy. Nejpomalejšího času dosáhl komunitní patch pro VZDump, který provádí detekci rozdílných dat vůči poslední plné záloze až při požadavku na vytvoření diferenciální zálohy, a na rozdíl od ostatních řešení nevyužívá dirty bitmapu. K tomuto účelu komunitní patch pro VZDump využívá nástroj `pve-xdelta3`, který při detekci rozdílných dat dosahuje výkonu pouze 5-13 MB/s, která je dána její výpočetní náročností. Ve fázi, kdy není nástrojem detekován rozdíl dat, tak je výkon tvorby zálohy shodný s naměřenými hodnotami v případě vytváření plné zálohy.

Vzhledem k rychlosti vytvoření záloh vyvinutým řešením a Proxmox Backup Serverem nebylo možné změřit pokles dostupného výkonu zápisu a čtení ve virtuálním stroji při tvorbě zálohy, jelikož toto dočasné vytížení disku bylo vykompenzováno pomocí I/O cache operačního systému ve virtuálním stroji. U komunitního patche pro VZDump není pokles dostupného výkonu uveden, jelikož se výrazně liší v každé z popisovaných fází, a jeho aritmetický průměr není vypovídající. Pokud je nástrojem `pve-xdelta3` detekován rozdíl v datech vůči poslední plné záloze, tak vzhledem k nízké rychlosti čtení disku tímto nástrojem není možné pokles dostupného výkonu pozorovat. V případě, že nástroj `pve-xdelta3` nedetekuje rozdílná data vůči poslední plné

záloze, tak je detekce rozdílu velmi rychlá a zálohovaný disk je čten téměř shodnou rychlostí jako v případě tvorby plné zálohy, což vede i ke shodnému omezení dostupného výkonu čtení a zápisu ve virtuálním stroji.

Zálohovací řešení	Čas
VZDump	nepodporuje
Proxmox Backup Server	15s
Komunitní patch pro VZDump	45m 32s
Vyvinuté řešení	16s

Tabulka 7.4: Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 100 GB se změnou 1 GB

Inkrementální záloha disku o kapacitě 100 GB se změnou dat od poslední plné zálohy vytvořilo vyvinuté řešení i Proxmox Backup Server v téměř shodném čase. Jak již bylo zmíněno, tak s delším časem zálohy dochází k větší zanedbatelnosti režie na vytvoření a smazání snímku disku, a proto se naměřený čas pro vytvoření zálohy 10 GB a 100 GB disku liší pouze o jednu sekundu. Pro Proxmox Backup Server byl vůči vytvoření zálohy 10 GB disku naměřen dvojnásobný čas, což je pravděpodobně dáno nižším výkonem analyzátoru dirty bitmapy než u vyvinutého řešení, kde měla velikost dirty bitmapy minimální vliv na celkový potřebný čas k vytvoření zálohy.

7.3.3 Vytvoření inkrementální/diferenciální zálohy (3 GB a 30 GB změna dat)

V rámci tohoto scénáře byl pro 10 GB disk změřen potřebný čas k vytvoření inkrementální či diferenciální zálohy se změnou 3 GB dat vůči poslední plné záloze. Pro disk o velikosti 100 GB byla měřena změna 30 GB dat vůči poslední plné záloze. Uvedené hodnoty změny dat odpovídají 30 % z celkové kapacity disku, a jejich účelem je reprezentovat dvě různé velikosti aktivně využívaného databázového serveru s velkým množství zápisových operací.

Zálohovací řešení	Čas zálohy bez měření výkonu	Čas zálohy při měření zápisu	Naměřený výkon zápisu ve VM (změna vůči referenčním hodnotám)	Čas zálohy při měření čtení	Naměřený výkon čtení ve VM (změna vůči referenčním hodnotám)
VZDump	nepodporuje				
Proxmox Backup Server	39s	39s	190 MB/s (-42,2 %)	39s	251 MB/s (-21,8 %)
Komunitní patch pro VZDump	28m 58s	neměřeno			
Vyvinuté řešení	40s	58s	190 MB/s (-42,2 %)	41s	188 MB/s (-41,4 %)

Tabulka 7.5: Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 10 GB se změnou 3 GB dat

Inkrementální zálohu disku o velikosti 10 GB se změnou 3 GB dat vůči poslední plné záloze vytvořilo vyvinuté řešení i Proxmox Backup Server v téměř shodném čase, což je dáno tím, že obě řešení využívají pro sledování změn dat dirty bitmapy. Nejpomaleji vytvořil diferenciální zálohu komunitní patch pro VZDump, což je opět dáno odlišnou metodou detekce změněných dat. U komunitního patche pro VZDump nebyl měřen dostupný výkon zápisu a čtení ve virtuálním stroji, jelikož se ve fázi detekce rozdílných dat výrazně liší od fáze, kdy jsou data shodná jako v poslední plné záloze (viz 7.3.2).

Zálohovací řešení	Čas zálohy bez měření výkonu	Čas zálohy při měření zápisu	Naměřený výkon zápisu ve VM (změna vůči referenčním hodnotám)	Čas zálohy při měření čtení	Naměřený výkon čtení ve VM (změna vůči referenčním hodnotám)
VZDump	nepodporuje				
Proxmox Backup Server	7m 15s	7m 3s	82 MB/s (-75,1 %)	7m 1s	221 MB/s (-31,2 %)
Komunitní patch pro VZDump	3h 48m	neměřeno			
Vyvinuté řešení	5m 18s	8m 49s	254 MB/s (-22,8 %)	5m 10s	208 MB/s (-35,2 %)

Tabulka 7.6: Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 100 GB se změnou 30 GB dat

V případě vytváření inkrementální zálohy 100 GB disku se změnou 30 GB dat bylo vyvinuté řešení nejlepší, jelikož se na delším časovém úseku opět projevil vyšší výkon tvorby samotné zálohy, čímž byla vykompenzována potřebná režie pro vytvoření a smazání snímku disku. Komunitní patch pro VZDump opět dosáhl nejhoršího času, což je dáno velmi pomalým výkonem tvorby diferenciální zálohy. Pokles dostupného výkonu zápisu a čtení ve virtuálním stroji nebyl u testu komunitního patche pro VZDump měřen ze shodného důvodu jako v předchozím případě.

7.3.4 Vytvoření inkrementální/diferenciální zálohy (bez změny dat)

Účelem testu je srovnat rychlost tvorby inkrementální či diferenciální zálohy, pokud na disku od poslední zálohy neproběhla žádná změna. Ačkoliv je tento případ v reálné praxi téměř nereálný, tak slouží především k porovnání odlišných metod pro detekci změněných dat od poslední zálohy.

Zálohovací řešení	Čas
VZDump	nepodporuje
Proxmox Backup Server	2s
Komunitní patch pro VZDump	2m 18s
Vyvinuté řešení	7s

Tabulka 7.7: Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 10 GB bez změny dat

Inkrementální zálohu disku o velikosti 10 GB bez změny dat vůči poslední plné záloze nejrychleji dokončil Proxmox Backup Server. Vyvinuté řešení bylo třikrát pomalejší, což je opět dáno režií pro vytvoření a smazání snímku disku (viz 7.3.1). Nejhorší čas byl naměřen opět u komunitního patche pro VZDump, což je dáno jeho neefektivní metodou pro detekci změněných dat (viz 7.3.2).

Zálohovací řešení	Čas
VZDump	nepodporuje
Proxmox Backup Server	2s
Komunitní patch pro VZDump	35m 23s
Vyvinuté řešení	6s

Tabulka 7.8: Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 100 GB bez změny dat

V případě vytvoření inkrementální zálohy disku o velikosti 100 GB bez změny dat vůči poslední plné záloze byl opět nejrychlejší Proxmox Backup Server, který dosáhl shodného času jako u disku o velikosti 10 GB. Téměř shodného času jako v případě vytvoření zálohy 10 GB disku dosáhlo i vyvinuté řešení, což je způsobeno konstantní režií pro vytvoření a smazání snímku zálohovaného disku. V rámci tohoto měření se nejvíce projevila neefektivita komunitního patche pro VZDump, který i v tomto případě musí přečíst celý zálohovaný disk, přestože nejsou žádná přečtená data použita do výsledné diferenciální zálohy, která je v tomto případě prázdná.

7.3.5 Vytvoření inkrementální/diferenciální zálohy (změna všech dat)

Podobně jako předchozí scénář, tak by ani tento neměl v reálné praxi nastat. Pokud k tomu došlo, tak je to důsledkem nevhodně zvolené frekvence plných záloh. Účelem tohoto měření je ukázat opačný extrém při používání odlišných metod pro detekci změněných dat od poslední zálohy.

Zálohovací řešení	Čas zálohy bez měření výkonu	Čas zálohy při měření zápisu	Naměřený výkon zápisu ve VM (změna vůči referenčním hodnotám)	Čas zálohy při měření čtení	Naměřený výkon čtení ve VM (změna vůči referenčním hodnotám)
VZDump	nepodporuje				
Proxmox Backup Server	2m 20s	2m 29s	107 MB/s (-67,5 %)	2m 34s	215 MB/s (-33,0 %)
Komunitní patch pro VZDump	1h 20m	neměřeno			
Vyvinuté řešení	1m 48s	2m 40s	122 MB/s (-62,9 %)	1m 52s	183 MB/s (-43,0 %)

Tabulka 7.9: Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 10 GB se změnou všech dat

Z tabulky 7.9 vyplývá, že při inkrementální záloze disku o velikosti 10 GB je nejrychlejší vyvinuté řešení, a dosahuje velmi podobného času jako v případě plné zálohy (viz 7.3.1). Pro Proxmox Backup Server byl naměřen o 43 % horší čas než v případě vytvoření plné zálohy, čímž se potvrzuje předchozí tvrzení o neefektivnosti jeho analyzátoru dirty bitmapy (viz 7.3.2). Nejhorší čas byl naměřen u komunitního patche pro VZDump, u kterého se rychlost vytváření zálohy pohybovala pouze mezi 1-3 MB/s, kvůli čemuž nebylo možné změřit ani pokles dostupného výkonu zápisu a čtení ve virtuálním stroji.

Zálohovací řešení	Čas zálohy bez měření výkonu	Čas zálohy při měření zápisu	Naměřený výkon zápisu ve VM (změna vůči referenčním hodnotám)	Čas zálohy při měření čtení	Naměřený výkon čtení ve VM (změna vůči referenčním hodnotám)
VZDump	nepodporuje				
Proxmox Backup Server	23m 34s	25m 39s	83 MB/s (-74,8 %)	25m 17s	203 MB/s (-36,8 %)
Komunitní patch pro VZDump	13h 46m	neměřeno			
Vyvinuté řešení	15m 32s	23m 21s	104 MB/s (-64,8 %)	15m 41s	159 MB/s (-50,5 %)

Tabulka 7.10: Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 100 GB se změnou všech dat

Při vytváření inkrementální zálohy disku o velikosti 100 GB se změnou všech dat bylo opět nejrychlejší vyvinuté řešení, které dokončilo zálohu v téměř shodném čase jako v případě vytváření plné zálohy (viz 7.3.1). Proxmox Backup Server vytvořil inkrementální zálohu o 10 % pomaleji než v případě plné zálohy, a neefektivita analyzátoru dirty bitmapy v tomto případě neměla tak výrazný vliv na celkový čas zálohy jako v předchozím případě. Komunitní patch pro VZDump opět vykázal nejhorší výsledky, což je způsobeno velmi pomalou tvorbou diferenciální zálohy. Díky nízkému výkonu tvorby zálohy opět nebyl dostupný výkon zápisu a čtení ve virtuálním stroji měřitelný.

7.3.6 Obnova dat z plné zálohy

Obnova dat z plné zálohy, podobně jako u vytvoření plné zálohy, reprezentuje základní případ využití zálohovacího řešení.

Zálohovací řešení	Čas
VZDump	1m 47s
Proxmox Backup Server	3m 38s
Komunitní patch pro VZDump	1m 41s
Vyvinuté řešení	1m 27s

Tabulka 7.11: Výsledek měření zálohovacích řešení pro obnovu dat z plné zálohy o velikosti 10 GB

V případě obnovy plné zálohy disku o velikosti 10 GB dosáhlo nejlepšího výsledku vyvinuté řešení, což je dáno tím, že je plná záloha ukládána jako obraz disku, a operace obnovy spočívá v pouhém kopírování dat. VZDump i jeho komunitní patch dosáhl horšího výsledku, jelikož pro uložení zálohy využívá vlastní formát `vma`, u kterého není obnova dat tak triviální operací. Nejhoršího času dosáhl Proxmox Backup Server, který ukládá zálohy jako velké množství souborů o velikostech v řádu jednotek megabajtů, což v delším časovém horizontu vede k ušetření úložného prostoru, jelikož znovu neukládá duplicitní segmenty dat, které se mohou vyskytovat i napříč více virtuálními stroji. Avšak nevýhodou je, že v případě požadavku na obnovu virtuálního stroje musí provést opětovné složení souborů obsahující zálohu jednotlivých segmentů dat, což vede k výraznému prodloužení času obnovy.

Zálohovací řešení	Čas
VZDump	17m 57s
Proxmox Backup Server	39m 6s
Komunitní patch pro VZDump	17m 1s
Vyvinuté řešení	18m 39s

Tabulka 7.12: Výsledek měření zálohovacích řešení pro obnovu dat z plné zálohy o velikosti 100 GB

Z tabulky 7.12 vyplývá, že v případě obnovy plné zálohy o velikosti 100 GB byl nejrychlejší VZDump i jeho komunitní patch, jelikož formát `vma` obsahuje informaci o tom, zda konkrétní blok dat o velikosti 4KiB obsahuje

pouze nuly, a v takovém případě nedochází ke čtení dat ze zálohy, ale k zápisu na cílové úložiště z připraveného konstantního pole. Vyvinuté řešení díky jednoduchosti formátu zálohy nemá informaci o nulových datech bez jejich přečtení, a tak nemůže podobnou optimalizaci využít. Tato nevýhoda vyvinutého řešení je vyvážena možností připojení plné zálohy pomocí standardních nástrojů GNU/Linux. Zálohu ve formátu `vma` není možné snadno připojit, a je nutné provést její předchozí obnovu do virtuálního stroje.

7.3.7 Obnova dat z inkrementální/diferenciální zálohy - (1 GB změna dat)

Tento testovací scénář reprezentuje obnovu dat z inkrementální či diferenciální zálohy, která obsahuje 1 GB změnu dat vůči poslední plné záloze. Cílem je prověření rychlosti obnovení zálohy, která vznikla na základě měření v kapitole 7.3.2.

Zálohovací řešení	Čas
VZDump	nepodporuje
Proxmox Backup Server	3m 27s
Komunitní patch pro VZDump	1m 51s
Vyvinuté řešení	1m 29s

Tabulka 7.13: Výsledek měření zálohovacích řešení pro obnovu dat z inkrementální/diferenciální zálohy o velikosti 10 GB se změnou 1 GB dat

Z provedených měření vyplývá, že inkrementální zálohu disku o velikosti 10 GB nejrychleji obnovilo vyvinuté řešení, přičemž je naměřený čas téměř shodný jako v případě obnovení plné zálohy. Čas komunitního patche pro VZDump byl mírně vyšší než u plné zálohy, jelikož neimplementuje tak efektivní sloučení plné a diferenciální zálohy jako vyvinutého řešení. Nejpomaleji zálohu obnovil Proxmox Backup Server, avšak naměřený výsledek je velmi podobný obnově plné zálohy, a potřeba sloučení plné a inkrementální zálohy neměla výrazný vliv na čas obnovy dat.

Zálohovací řešení	Čas
VZDump	nepodporuje
Proxmox Backup Server	39m 27s
Komunitní patch pro VZDump	16m 32s
Vyvinuté řešení	18m 34s

Tabulka 7.14: Výsledek měření zálohovacích řešení pro obnovu dat z inkrementální/diferenciální zálohy o velikosti 100 GB se změnou 1 GB dat

V případě obnovy diferenciální zálohy disku o velikosti 100 GB dosáhl nejlepšího času komunitní patch pro VZDump, jelikož využívá shodné optimalizace jako samotný VZDump, kdy nejsou nulová data čtena ze zálohy disku (viz 7.3.6). Čas vyvinutého řešení byl znovu velmi podobný času obnovy plné zálohy, jelikož provádí sloučení záloh velmi efektivní metodou, a toto sloučení nemá pozorovatelný vliv na celkový čas provedení zálohy.

7.3.8 Obnova dat z inkrementální/diferenciální zálohy - (3 GB a 30 GB změna dat)

Posledním testovacím scénářem je obnova disku z inkrementální nebo diferenciální zálohy, která obsahuje změnu 3 GB dat v případě 10 GB disku a změnu 30 GB dat v případě 100 GB disku. Cílem je změření rychlosti obnovení zálohy, která vznikla na základě měření v kapitole 7.3.3.

Zálohovací řešení	Čas
VZDump	nepodporuje
Proxmox Backup Server	3m 32s
Komunitní patch pro VZDump	2m 23s
Vyvinuté řešení	1m 30s

Tabulka 7.15: Výsledek měření zálohovacích řešení pro obnovu dat z inkrementální/diferenciální zálohy o velikosti 10 GB se změnou 3 GB dat

Z tabulky 7.15 vyplývá, že vyvinuté řešení bylo v úloze obnovení inkrementální zálohy disku o velikosti 10 GB se změnou 3 GB dat nejlepší,

přičemž byl celkový čas obnovy opět velmi podobný času obnovy plné zálohy (viz 7.3.6).

Zálohovací řešení	Čas
VZDump	nepodporuje
Proxmox Backup Server	40m 57s
Komunitní patch pro VZDump	28m 39s
Vyvinuté řešení	18m 35s

Tabulka 7.16: Výsledek měření zálohovacích řešení pro obnovu dat z inkrementální/diferenciální zálohy o velikosti 10 GB se změnou 3 GB dat

Inkrementální zálohu disku o velikosti 100 GB se změnou 30 GB dat nejrychleji obnovilo vyvinuté řešení. V tomto případě dosáhl komunitní patch pro VZDump horšího času než u obnovy zálohy disku o kapacitě 100 GB se změnou 1 GB dat, což je způsobenou neefektivitou sloučení více záloh.

7.4 Shrnutí

Z provedených měření vyplývá, že je vyvinuté řešení konkurenceschopné, a ve většině úloh dosahuje srovnatelných či lepších výsledků než oficiální Proxmox Backup Server. Horších výsledků dosahuje především v případě, že vytvoření zálohy trvá pouze jednotky sekund, kde se nejvíce projeví konstantní režie pro vytvoření a následné smazání snímku disku.

Dále vyvinuté řešení nejvíce omezuje dostupný výkon čtení ve virtuálním stroji, jelikož nevyužívá podobnou optimalizaci jako ostatní popisovaná řešení, kdy jsou čtecí operace QEMU využívány zároveň pro virtuální stroj i účely vytvoření zálohy, díky čemuž je z pohledu disku provedena jen jedna čtecí operace. Avšak tato optimalizace by vyžadovala vývoj vlastního formátu zálohy, který by umožňoval uložení dat v náhodném pořadí. Současně vyvinuté řešení ukládá plnou zálohu jako obraz disku, který je pro účely procházení možné připojit pomocí standardních nástrojů GNU/Linux, což by s vlastním formátem zálohy nebylo možné.

8 Závěr

Cílem práce bylo implementovat systém pro vytváření inkrementálních záloh virtuálních strojů provozovaných za pomoci QEMU/KVM. K dosažení požadovaného výsledku bylo zapotřebí splnit dva dílčí cíle.

Prvním z nich bylo rozšíření implementace dirty bitmap v QEMU (viz 5.3), které zahrnovalo přidání podpory jejich persistence pro virtuální disky ve formátu RAW a rozšíření rozhraní QMP API o příkaz zprostředkovávající přístup k obsahu dirty bitmapy. Všechny změny v projektu QEMU byly implementovány s důrazem na jejich obecnost bez přímé návaznosti na další cíle práce, čímž je umožněno využívání těchto změn dalšími projekty. Splnění tohoto cíle proběhlo v plném rozsahu.

Druhým cílem bylo vyvinutí vlastního nástroje pro vytváření plných a inkrementálních záloh v programovacím jazyce Python (viz 5.4). Podstatným požadavkem byla podpora pro zálohu disků využívající síťové úložiště LINSTOR, které je využíváno Katedrou informatiky a výpočetní techniky. Nástroj vyžaduje, aby zálohované disky virtuálního využívaly úložiště LINSTOR či LVM z důvodu nutnosti vytváření jejich snímků, což vede k výraznému zlepšení konzistence zálohy. Dalším požadavkem bylo zachování možnosti živé migrace virtuálního stroje i po nasazení vyvinutého nástroje. Tato funkcionalita nebyla vyvinutým řešením nijak ovlivněna, a je možné ji s vyvinutým řešením nadále využívat (viz 5.4.3). Vyvinutý nástroj splňuje všechny požadavky a je možné ho plně využívat s hypervizorem QEMU/KVM.

Následně bylo celé řešení nad rámec zadání úspěšně integrováno do virtualizační platformy Proxmox Virtual Environment a její webové administrace (viz 5.5), ze které lze vyvinuté řešení plně spravovat.

Vyvinuté řešení bylo řádně otestováno v kapitole 7, a z výsledku měření vyplývá, že je při tvorbě zálohy průměrně o 30 % rychlejší než Proxmox Backup Server, a o 116 % rychlejší při její obnově. Pro účely měření bylo řešení experimentálně nasazeno na jednom ze serverů Katedry informatiky a výpočetní techniky, kde bude i nadále využíváno.

Práce může být v budoucnu rozšířena o možnost vytváření diferenciálních záloh nebo vytvoření virtuálního blokového zařízení z existujících inkrementálních záloh, což by umožnilo procházet jejich obsah bez nutnosti předchozího sestavení obrazu disku do samostatného souboru.

Tato práce splňuje všechny body zadání, a výsledné softwarové řešení je vhodné k běžnému používání za účelem ochrany dat před ztrátou nebo poškozením vlivem chyby hardwaru, softwaru i uživatele.

Přehled použitých zkratek

AGPL	Affero General Public License
AMD-V	AMD Virtualization
AMD	Advanced Micro Devices
API	Application Programming Interface
ARM	Advanced RISC Machine
AVHD(X)	Automatic Virtual Hard Disk
AWS	Amazon Web Services
btrfs	B-tree file system
CD	Compact Disc
CPU	Central Processing Unit
DCUI	Direct Console User Interface
DDR3	Double Data Rate 3
DEP	Data Execution Prevention
DRBD	Distributed Replicated Block Device
EC2	Elastic Compute Cloud
ext2	Second extended file system
ext3	Third extended file system
ext4	Fourth extended file system
FAT	File Allocation Table
GNU	GNU's Not Unix!
GPL	General Public License
HDD	Hard Disk
HFS	Hierarchical File System
HTML5	HyperText Markup Language 5
ISO	International Organization for Standardization
I/O	Input/Output

Intel VT	Intel Virtualization Technology
JFS	Journaled File System
JSON	JavaScript Object Notation
KVM	Kernel-based Virtual Machine
LVM	Logical Volume Manager
LXC	Linux Containers
NBD	Network Block Device
NFS	Network File System
NILFS	New Implementation of a Log-structured File System
NTFS	New Technology File System
OS	Operační systém
PBS	Proxmox Backup Server
PIP	Pip Installs Packages
PVE	Proxmox Virtual Environment
Proxmox VE	Proxmox Virtual Environment
QAPI	QMP API
QCOW	QEMU Copy On Write
QCOW2	QEMU Copy On Write 2
QED	QEMU Enhanced Disk
QEMU	Quick Emulator
QMP	QEMU Monitor Protocol
RAID	Redundant Array of Inexpensive Disks
RAM	Random Access Memory
RDP	Remote Desktop Protocol
RHEV	Red Hat Enterprise Virtualization
RHV	Red Hat Virtualization
RISC	Reduced Instruction Set Computer
S3	Simple Storage Service
SCSI	Small Computer System Interface

SLAT	Second Level Address Translation
SPICE	Simple Protocol for Independent Computing Environments
SSO	Single Sign-On
vCPU	Virtual CPU
VDI	Virtual Disk Image
VHD(X)	Virtual Hard Disk
VM	Virtual Machine (virtuální stroj)
VMDK	Virtual Machine Disk
VMM	Virtual Machine Monitor
VMware ESXi	VMware Elastic Sky X Integrated
VUL	VMware Universal License
ZFS	Zettabyte File System

Seznam obrázků

3.1 Protection ring v případě využití virtualizovaného prostředí s hardwarovou podporou virtualizace 20
3.2 Architektura hypervizoru prvního typu 21
3.3 Architektura hypervizoru druhého typu 21
3.4 Administrační rozhraní Oracle VM VirtualBox 27
3.5 Administrační rozhraní Parallels Desktop [23] 28
3.6 Administrační rozhraní Hyper-V Manager [33] 31
3.7 Administrační rozhraní Windows Admin Center [34] 32
3.8 Administrační rozhraní HV Manager [35] 32
3.9 Administrační rozhraní VMware ESXi 34
3.10 Administrační rozhraní Proxmox Virtual Environment . . . 35

5.1 Struktura provedených změn v projektu QEMU 47
5.2 Diagram modulů nástroje pro vytváření a obnovu záloh . . . 51
5.3 Vizualizace vzniku nekonzistentní zálohy bez získání obsahu dirty bitmapy a vytvoření snímku disku v rámci jedné atomické operace 57
5.4 Příklad virtuálního disku vytvořeného k obnově 58
5.5 Upravený seznam provedených záloh 61

6.1 Modifikovaný formulář pro vytvoření virtuálního stroje na platformě Proxmox VE 65
6.2 Modifikovaný formulář pro definici disku při vytváření virtuálního stroje v platformě Proxmox VE 66

Seznam ukázek kódu

5.1 Konfigurace bitmapy pomocí parametrů QEMU 49
5.2 konfigurace v případě použití s Proxmox VE a LINSTOR . . 53
5.3 Vytvoření inkrementální zálohy virtuálního stroje 57
5.4 Obnova zálohy do původního virtuálního disku 58
6.1 Ukázka zkopírování instalačního souboru QEMU na server . 62
6.2 Ukázka instalace upravené verze QEMU 63
6.3 Ověření správné kompilace a instalace `pve-qemu` 63
6.4 Instalace nástroje pro vytváření záloh 64
6.5 Ukázka úspěšné aplikace patche 64
7.1 Příkaz použitý pro zaplnění celého disku náhodnými daty . . 68
7.2 Skript použitý pro zaplnění části disku náhodnými daty . . . 69
7.3 Příkazy použité pro měření výkonu zápisu a čtení 69
B.1 Ukázka zkopírování a rozbalení upravené verze QEMU . . .
B.2 Ukázka instalace upravené verze QEMU . . . . . .
C.1 Příkaz pro smazání vybrané zálohy . . . . . .
C.2 Přidání nového virtuálního stroje do konfiguračního souboru
C.3 Přidání nového virtuálního disku do konfiguračního souboru

Seznam tabulek

2.1 Souhrn parametrů popsaných typů záloh 16

3.1 Souhrn parametrů popsaných typů záloh 36

4.1 Srovnání zálohovacích řešení pro virtuální stroje 44

7.1 Výsledek měření zálohovacích řešení pro plnou zálohu disku o velikosti 10 GB 71
7.2 Výsledek měření zálohovacích řešení pro plnou zálohu disku o velikosti 100 GB 72
7.3 Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 10 GB se změnou 1 GB . . 73
7.4 Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 100 GB se změnou 1 GB . 74
7.5 Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 10 GB se změnou 3 GB dat 75
7.6 Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 100 GB se změnou 30 GB dat 76
7.7 Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 10 GB bez změny dat . . . 77
7.8 Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 100 GB bez změny dat . . 77
7.9 Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 10 GB se změnou všech dat 78
7.10 Výsledek měření zálohovacích řešení pro inkrementální/diferenciální zálohu disku o velikosti 100 GB se změnou všech dat 79
7.11 Výsledek měření zálohovacích řešení pro obnovu dat z plné zálohy o velikosti 10 GB 80
7.12 Výsledek měření zálohovacích řešení pro obnovu dat z plné zálohy o velikosti 100 GB 80
7.13 Výsledek měření zálohovacích řešení pro obnovu dat z inkrementální/diferenciální zálohy o velikosti 10 GB se změnou 1 GB dat 81

7.14 Výsledek měření zálohovacích řešení pro obnovu dat z inkrementální/diferenciální zálohy o velikosti 100 GB se změnou 1 GB dat 82
7.15 Výsledek měření zálohovacích řešení pro obnovu dat z inkrementální/diferenciální zálohy o velikosti 10 GB se změnou 3 GB dat 82
7.16 Výsledek měření zálohovacích řešení pro obnovu dat z inkrementální/diferenciální zálohy o velikosti 10 GB se změnou 3 GB dat 83

Literatura

[1] Humble Devassy Chirammal, A. V. P. M. *Mastering KVM Virtualization*. Packt Publishing, 2016. ISBN 9781784399054.

[2] Both, D. *Using and Administering Linux: Volume 2: Zero to SysAdmin: Advanced Topics*. Apress, 2019. ISBN 9781484254554.

[3] *Block Storage Management For Containers* [online]. LINBIT, 2021. [cit. 2021/04/03]. Dostupné z: `https://linbit.com/linstor/`.

[4] Ahmed, W. *Mastering Proxmox - Third Edition*. Packt Publishing, 2017. ISBN 9781788397605.

[5] Nelson, S. *Pro Data Backup and Recovery*. Apress, 2011. ISBN 978-1-4302-2663-5.

[6] Collins, T. *Full Backup vs. Incremental Backup vs. Differential Backup: Which Is Best?* [online]. Atlantech Online, Inc., 2021. [cit. 2021/02/17]. Dostupné z: `https://www.atlantech.net/blog/full-backup-vs.-incremental-backup-vs.-differential-backup-which-is-best`.

[7] Techforce, D. *What's The Difference Between File Level and Image Level Backup?* [online]. Darby Techforce, 2021. [cit. 2021/02/17]. Dostupné z: `https://darbytechforce.com/uncategorized/whats-the-difference-between-file-level-and-image-level-backup/`.

[8] *SysadminGuide* [online]. 2021. [cit. 2021/02/18]. Dostupné z: `https://btrfs.wiki.kernel.org/index.php/SysadminGuide`.

[9] Wojsław, D. *Introducing ZFS on Linux: Understand the Basics of Storage with ZFS*. Apress, 2017. ISBN 9781484233061.

[10] *Ext4 (and Ext2/Ext3) Wiki* [online]. 2021. [cit. 2021/03/30]. Dostupné z: `https://ext4.wiki.kernel.org/index.php/Main_Page`.

[11] Rodríguez-Haro, F. et al. A summary of virtualization techniques. *Procedia Technology*. 12 2012, 3, s. 267–272. doi: 10.1016/j.protcy.2012.03.029.

[12] Chris Takemura, L. S. C. *The Book of Xen: A Practical Guide for the System Administrator (1st Edition)*. No Starch Press, 2009. ISBN 978-1593271862.

[13] Graziano, C. D. *A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project* [online]. Iowa State University, 2011. [cit. 2021/02/18]. Dostupné z: `https://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=3243&context=etd`.

[14] Meier, S. *IBM Systems Virtualization: Servers, Storage, and Software.* IBM Redpaper publication, 2008.

[15] *ESXi* [online]. VMware, Inc., 2021. [cit. 2021/04/01]. Dostupné z: `https://www.vmware.com/products/esxi-and-esx.html`.

[16] *Introduction to Hyper-V on Windows 10* [online]. Microsoft, 2021. [cit. 2021/03/29]. Dostupné z: `https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/`.

[17] Colvin, H. *VirtualBox: An Ultimate Guide Book on Virtualization with VirtualBox*. CreateSpace Independent Publishing Platform, 2015. ISBN 978-1522769880.

[18] *Distributed Replicated Storage System* [online]. LINBIT, 2021. [cit. 2021/04/27]. Dostupné z: `https://linbit.com/drbd/`.

[19] affiliates, O. *Disk Image Files (VDI, VMDK, VHD, HDD)* [online]. Oracle and/or its affiliates, 2020. [cit. 2021/04/08]. Dostupné z: `https://docs.oracle.com/en/virtualization/virtualbox/6.0/user/vdidetails.html`.

[20] McLoughlin, M. *The QCOW Image Format* [online]. Mark McLoughlin, 2006. [cit. 2021/04/08]. Dostupné z: `https://people.gnome.org/~markmc/qcow-image-format-version-1.html`.

[21] *Features/QED* [online]. QEMU, 2017. [cit. 2021/04/08]. Dostupné z: `https://wiki.qemu.org/Features/QED`.

[22] Kissell, J. *Take Control of Parallels Desktop 12*. O'Reilly Media, Inc., 2016. ISBN 9781615424740.

[23] Bookwalter, J. *Parallels Desktop 16 for Mac review: Standing at the crossroads of Mac's future* [online]. IDG Communications, Inc., 2020. [cit. 2021/04/27]. Dostupné z: `https://www.macworld.com/article/234539/parallels-desktop-16-for-mac-review.html`.

[24] *The GNU General Public License* [online]. Free Software Foundation, Inc., 2021. [cit. 2021/03/29]. Dostupné z: `https://www.gnu.org/licenses/licenses.html#GPL`.

[25] *macOS Big Sur* [online]. Apple Inc., 2021. [cit. 2021/03/29]. Dostupné z: https://www.apple.com/cz/macos/big-sur/.

[26] *QEMU Machine Protocol* [online]. QEMU, 2019. [cit. 2021/02/19]. Dostupné z: https://wiki.qemu.org/Documentation/QMP.

[27] *libvirt - Virtualization API* [online]. libvirt, 2019. [cit. 2021/02/19]. Dostupné z: https://libvirt.org/.

[28] *Using the KVM API* [online]. Eklektix, Inc., 2021. [cit. 2021/02/19]. Dostupné z: https://lwn.net/Articles/658511/.

[29] CONTRIBUTORS, K. *Information about what cpu supports Hardware virtualization* [online]. KVM, 2014. [cit. 2021/02/19]. Dostupné z: https://www.linux-kvm.org/page/Processor_support.

[30] *Microsoft* [online]. Microsoft, 2021. [cit. 2021/03/29]. Dostupné z: https://microsoft.com/en-us/.

[31] *Microsoft Windows* [online]. Microsoft, 2021. [cit. 2021/03/29]. Dostupné z: https://www.microsoft.com/en-us/windows.

[32] *System requirements for Hyper-V on Windows Server* [online]. Microsoft, 2021. [cit. 2021/02/19]. Dostupné z: https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/system-requirements-for-hyper-v-on-windows.

[33] POSEY, B. *Retrieving Detailed Information About Hyper-V VMs* [online]. 1105 Media Inc., 2018. [cit. 2021/04/27]. Dostupné z: https://redmondmag.com/articles/2018/08/08/retrieving-detailed-info-about-hyper-v-vms.aspx.

[34] *Managing Virtual Machines with Windows Admin Center* [online]. Microsoft, 2021. [cit. 2021/02/19]. Dostupné z: https://docs.microsoft.com/en-us/windows-server/manage/windows-admin-center/use/manage-virtual-machines.

[35] *Hv Manager features* [online]. AprelTech, 2021. [cit. 2021/02/19]. Dostupné z: https://hv-manager.org/#features.

[36] *ESXi Hardware Requirements* [online]. VMware, Inc., 2021. [cit. 2021/02/19]. Dostupné z: https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.esxi.install.doc/GUID-DEB8086A-306B-4239-BF76-E354679202FC.html.

[37] *VMware Compatibility Guide* [online]. VMware, Inc., 2021. [cit. 2021/02/19]. Dostupné z: https://www.vmware.com/resources/compatibility/search.php.

[38] VMware, I. *VMware Virtual Disks* [online]. VMware, Inc., 2007. [cit. 2021/04/05]. Dostupné z: https://www.vmware.com/app/vmdk/?src=vmdk.

[39] *About the Direct Console ESXi Interface* [online]. VMware, Inc., 2021. [cit. 2021/02/19]. Dostupné z: https://docs.vmware.com/en/VMware-vSphere/6.5/com.vmware.vsphere.install.doc/GUID-E64B4311-11E5-44E8-8DB5-B438B4A14289.html.

[40] *vSphere vMotion* [online]. VMware, Inc., 2021. [cit. 2021/03/29]. Dostupné z: https://www.vmware.com/products/vsphere/vmotion.html.

[41] *How does single sign-on work?* [online]. OneLogin, Inc., 2021. [cit. 2021/03/29]. Dostupné z: https://www.onelogin.com/learn/how-single-sign-on-works.

[42] *vCenter Server* [online]. VMware, Inc, 2021. [cit. 2021/02/19]. Dostupné z: https://www.vmware.com/products/vcenter-server.html.

[43] *debian - The universal operating system* [online]. Software in the Public Interest, Inc. and others, 2021. [cit. 2021/03/30]. Dostupné z: https://www.debian.org/.

[44] *System Requirements* [online]. Proxmox Server Solutions GmbH, 2021. [cit. 2021/02/19]. Dostupné z: https://www.proxmox.com/en/proxmox-ve/requirements.

[45] *Firefox Browser* [online]. Mozilla Corporation, 2021. [cit. 2021/03/30]. Dostupné z: https://www.mozilla.org/cs/firefox/new/.

[46] *Google Chrome* [online]. Software in the Public Interest, Inc. and others, 2021. [cit. 2021/03/30]. Dostupné z: https://www.google.com/intl/cs_CZ/chrome/.

[47] *SPICE* [online]. 2021. [cit. 2021/03/30]. Dostupné z: https://www.spice-space.org/index.html.

[48] Berrangé, D. P. *Virtual Machine Manager - Virt Viewer* [online]. Daniel P. Berrangé, 2021. [cit. 2021/03/30]. Dostupné z: https://virt-manager.org/download/.

[49] Both, D. *Mastering Veeam Backup & Replication 10*. Packt Publishing, 2021. ISBN 9781838980443.

[50] Martin Gavanda, P. V. K. N. A. M. *Mastering VMware vSphere 6.7 - Second Edition*. Packt Publishing, 2019. ISBN 9781789613377.

[51] Foulds, I. *Learn Azure in a Month of Lunches, Second Edition*. Manning Publications, 2020. ISBN 9781617297625.

[52] Wilkins, M. *Learning Amazon Web Services (AWS): A Hands-On Guide to the Fundamentals of AWS Cloud*. Addison-Wesley Professional, 2019. ISBN 9780135301104.

[53] *FAT File Systems. FAT32, FAT16, FAT12* [online]. 2021. [cit. 2021/04/02]. Dostupné z: `https://www.ntfs.com/fat_systems.htm`.

[54] Redbooks, I. *Hierarchical File System Usage Guide*. IBM, 2000. ISBN 978-0738419213.

[55] Mainyu, E. A. *JFS (File System): Journaling file system, eComStation, IBM AIX*. Aud Publishing, 2013. ISBN 978-620-0-94420-7.

[56] Lambert M. Surhone, S. F. M. M. T. T. *NILFS: Log-structured File System*. Betascript Publishing, 2010. ISBN 978-613-0-49330-1.

[57] Miller, F. P. – Vandome, A. F. – McBrewster, J. *NTFS*. Alpha Press, 2009. ISBN 6130077963.

[58] Lambert M. Surhone, S. F. H. M. T. T. *Hans Reiser: Namesys, ReiserFS, Reiser4, SUSE Linux, ARDC, Synopsys, Honda CRX*. Betascript Publishing, 2010. ISBN 978-613-3-13560-4.

[59] Sobell, M. G. *A Practical Guide to Fedora and Red Hat Enterprise Linux (7th Edition)*. Prentice Hall Press, 7th edition, 2014. ISBN 0133477436.

[60] *System requirements* [online]. Veeam Software, 2021. [cit. 2021/02/20]. Dostupné z: `https://www.veeam.com/backup-replication-system-requirements.html`.

[61] *How Backup Works* [online]. Veeam Software, 2021. [cit. 2021/02/20]. Dostupné z: `https://www.veeam.com/veeam_vbr_one_10_editions_comparison_ds.pdf`.

[62] *Instant VM Recovery* [online]. Veeam Software, 2021. [cit. 2021/02/20]. Dostupné z: `https://helpcenter.veeam.com/docs/backup/hyperv/instant_recovery.html?ver=100`.

[63] *Instant VM Recovery* [online]. Veeam Software, 2021. [cit. 2021/02/20]. Dostupné z: `https://helpcenter.veeam.com/docs/backup/vsphere/instant_recovery.html?ver=100`.

[64] *Entire VM Restore* [online]. Veeam Software, 2021. [cit. 2021/02/20]. Dostupné z: https://helpcenter.veeam.com/docs/backup/hyperv/full_recovery.html?ver=100.

[65] *Entire VM Restore* [online]. Veeam Software, 2021. [cit. 2021/02/20]. Dostupné z: https://helpcenter.veeam.com/docs/backup/vsphere/full_recovery.html?ver=100.

[66] *VM Files Restore* [online]. Veeam Software, 2021. [cit. 2021/02/20]. Dostupné z: https://helpcenter.veeam.com/docs/backup/hyperv/vmfile_recovery.html?ver=100.

[67] *VM Files Restore* [online]. Veeam Software, 2021. [cit. 2021/02/20]. Dostupné z: https://helpcenter.veeam.com/docs/backup/vsphere/vmfile_recovery.html?ver=100.

[68] *Feature Comparison* [online]. Veeam Software, 2021. [cit. 2021/02/20]. Dostupné z: https://www.veeam.com/products-edition-comparison.html.

[69] *Univerzální licence Veeam (VUL)* [online]. Veeam Software, 2021. [cit. 2021/03/30]. Dostupné z: https://www.veeam.com/cz/universal-licensing.html.

[70] *Veeam ONE - Účinné monitorování a analýza pro vaše prostředí IT* [online]. Veeam Software, 2021. [cit. 2021/03/30]. Dostupné z: https://www.veeam.com/cz/virtualization-management-one-solution.html.

[71] *Pricing and Packaging* [online]. Veeam Software, 2021. [cit. 2021/01/24]. Dostupné z: https://www.veeam.com/backup-solution-pricing.html.

[72] *Proxmox Backup Server* [online]. Proxmox Server Solutions GmbH, 2021. [cit. 2021/02/20]. Dostupné z: https://www.proxmox.com/en/proxmox-backup-server.

[73] *Roadmap* [online]. Proxmox Server Solutions GmbH, 2021. [cit. 2021/02/20]. Dostupné z: https://www.proxmox.com/en/news/press-releases?view=category&id=11.

[74] *Proxmox Backup Server Datasheet* [online]. Proxmox Server Solutions GmbH, 2021. [cit. 2021/02/20]. Dostupné z: https://www.proxmox.com/de/downloads/item/proxmox-backup-server-datasheet.

[75] *The GNU General Public License* [online]. Free Software Foundation, Inc., 2021. [cit. 2021/03/30]. Dostupné z: https://www.gnu.org/licenses/licenses.html#AGPL.

[76] *Proxmox VE 6.3 with Proxmox Backup Server Integration and Ceph Octopus released* [online]. Proxmox Server Solutions GmbH, 2021. [cit. 2021/02/20]. Dostupné z: https://www.proxmox.com/en/proxmox-backup-server/pricing.

[77] Preston, W. *Backup & Recovery*. O'Reilly Media, Inc., 2007. ISBN 9780596102463.

[78] Subramanian, P. *Getting Started with Red Hat Enterprise Virtualization*. Packt Publishing, 2014. ISBN 9781782167402.

[79] Jones, R. W. *guestfish - the guest filesystem shell* [online]. Richard W.M. Jones, 2020. [cit. 2021/04/07]. Dostupné z: https://libguestfs.org/guestfish.1.html.

[80] *Bacula Enterprise Datasheet* [online]. Bacula Systems, 2018. [cit. 2021/04/07]. Dostupné z: https://www.baculasystems.com/wp-content/uploads/bacula-enterprise-virtualization-datasheet_s.pdf.

[81] *Enterprise Edition Comparison with Community Version* [online]. Bacula Systems, 2021. [cit. 2021/04/07]. Dostupné z: https://www.baculasystems.com/corporate-data-backup-software-solutions/professional-backup-software/enterprise-community-comparison/.

[82] *Acronis Cyber Backup* [online]. Acronis International GmbH, 2021. [cit. 2021/02/20]. Dostupné z: https://www.acronis.cz/produkt/acronis-backup/.

[83] Whalen, E. *Oracle VM Implementation and Administration Guide*. Oracle Press, 2011. ISBN 9780071639200.

[84] Reed, M. *Mastering Citrix® XenServer®*. Packt Publishing, 2014. ISBN 978-1783287390.

[85] *16577: Acronis Cyber Protect, Acronis Cyber Backup: agent-based and agentless backup of virtual machines* [online]. Acronis International GmbH, 2021. [cit. 2021/02/20]. Dostupné z: https://kb.acronis.com/content/16577.

[86] *Acronis Backup 12.5* [online]. Acronis International GmbH, 2021. [cit. 2021/02/20]. Dostupné z: https://www.acronis.cz/wp-content/uploads/2016/10/AcronisBackup_12.5_userguide_en-US.pdf.

[87] *62585: Acronis Cyber Backup: KVM support* [online]. Acronis International GmbH, 2021. [cit. 2021/02/20]. Dostupné z: https://kb.acronis.com/content/62585.

[88] *Oracle VM Server for x86* [online]. Acronis International GmbH, 2021. [cit. 2021/02/20]. Dostupné z: https://www.acronis.com/en-us/solutions/backup/oracle-vm/features/.

[89] *Acronis Cyber Backup for Red Hat Virtualization and Kernel-based Virtual Machine servers* [online]. Acronis International GmbH, 2021. [cit. 2021/02/20]. Dostupné z: https://www.acronis.com/en-us/solutions/backup/rhv/.

[90] Breuer, P. T. *The Network Block Device* [online]. Slashdot Media, LLC, 2000. [cit. 2021/04/03]. Dostupné z: https://www.linuxjournal.com/article/3778.

[91] *Proxmox Virtual Environment - Installation* [online]. Proxmox Server Solutions GmbH, 2020. [cit. 2021/04/02]. Dostupné z: https://pve.proxmox.com/wiki/Installation.

[92] *lvcreate(8) - Linux man page* [online]. die.net, 2021. [cit. 2021/04/04]. Dostupné z: https://linux.die.net/man/8/lvcreate.

[93] Trzciński, K. *Proxmox Patches* [online]. GitHub, Inc., 2020. [cit. 2021/04/16]. Dostupné z: https://github.com/ayufan/pve-patches.

[94] *PVE Xdelta 3* [online]. GitHub, Inc., 2015. [cit. 2021/04/16]. Dostupné z: https://github.com/ayufan/pve-xdelta3.

A Obsah přiloženého CD

K práci je přiloženo CD s následujícím obsahem:

- **check-qemu-install.sh** - Skript pro kontrolu správné kompilace a instalace upravené verze QEMU.
- **docs** - Dokumentace ve formátu PDF a zdrojové soubory pro kompilaci pomocí LaTeX.
- **pve.patch** - Patch pro Proxmox VE.
- **pve-qemu.zip** - Upravená verze QEMU připravená pro kompilaci a instalaci v prostředí Proxmox VE.
- **pve-qemu-deps.sh** - Příkaz pro instalaci potřebných závislostí pro kompilaci upravené verze QEMU.
- **pve-qemu-kvm_5.2.0-5_amd64.deb** - Předkompilovaný instalační balíček pve-qemu.
- **qemu.zip** - Upravená verze QEMU, která může být samostatně zkompilována bez Proxmox VE
- **qemu-backuper** - Nástroj pro vytváření záloh.

B Návod pro vlastní kompilaci upravené verze QEMU pro Proxmox Virtual Environment

Z kořenového adresáře přiloženého CD pomocí nástroje `scp` zkopírujeme na Proxmox VE server připravený archiv `pve-qemu.zip`, který obsahuje repozitář, ze kterého je možné spustit samotnou kompilaci. Následně je potřeba se připojit na server a rozbalit tento archiv pomocí nástroje `unzip`.

```
user@local:/media/cdrom# scp pve-qemu.zip root@192.168.1.131:/root
pve-qemu.zip                      100%   416MB   6.3MB/s   01:42
user@local:/media/cdrom# ssh root@192.168.1.131
root@pve:~# unzip pve-qemu.zip
Archive:  pve-qemu.zip
   creating: pve-qemu/
  inflating: pve-qemu/backup.txt
... (zkráceno)
```

Ukázka kódu B.1: Ukázka zkopírování a rozbalení upravené verze QEMU

Nyní je potřeba nainstalovat závislosti, které jsou potřeba pro samotnou kompilaci QEMU. Vzhledem k jejich vysokému počtu je připravený instalační příkaz na přiloženém CD v souboru `pve-qemu-deps.sh`. Po nainstalování závislostí je možné se přesunout do složky `pve-qemu`, a spustit kompilaci pomocí nástroje `make`. Výstupem kompilace je soubor `pve-qemu-kvm_5.2.0-5_amd64.deb`, který je možné nainstalovat nástrojem `dpkg`. Po úspěšné instalaci je nutné pro aplikování změn restartovat služby `pvedaemon` a `pveproxy` pomocí nástroje `systemctl`.

```
user@local:/media/cdrom# scp pve-qemu-deps.sh root@192.168.1.131:/root
root@pve:~# bash pve-qemu-deps.sh
Reading package lists... Done
Building dependency tree
Reading state information... Done
... (zkráceno)
root@pve:~# cd pve-qemu
root@pve:~/pve-qemu# make
```

```
test -f "qemu/configure" || git submodule update --init --
    ↪ recursive
cd pve-qemu-kvm-5.1.0; dpkg-buildpackage -b -us -uc -j
dpkg-buildpackage: info: source package pve-qemu-kvm
dpkg-buildpackage: info: source version 5.2.0-5
... (zkráceno)
dpkg-buildpackage: info: binary-only upload (no source
    ↪ included)
lintian pve-qemu-kvm_5.2.0-5_amd64.deb pve-qemu-kvm-dbg_5
    ↪ .2.0-5_amd64.deb
root@pve:~/pve-qemu# dpkg -i pve-qemu-kvm*.deb
(Reading database ... 121632 files and directories currently
    ↪ installed.)
Preparing to unpack pve-qemu-kvm_5.2.0-5_amd64.deb ...
Unpacking pve-qemu-kvm (5.2.0-5) over (5.2.0-5) ...
Setting up pve-qemu-kvm (5.2.0-5) ...
Processing triggers for mime-support (3.62) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for man-db (2.8.5-2) ...
root@pve:~/pve-qemu# systemctl restart pvedaemon pveproxy
root@pve:~/pve-qemu#
```

Ukázka kódu B.2: Ukázka instalace upravené verze QEMU

V aktuální fázi je nainstalována upravená verze QEMU implementující potřebné změny. Úspěšnost instalace ověříme stejným způsobem jako v kapitole 6.1.1.

C Uživatelská příručka k nástroji pro vytváření a obnovu záloh

Vyvinutý nástroj implementuje příkazy pro vytvoření plné a inkrementální zálohy, obnovení zálohy, smazání zálohy a konfiguraci virtuálních strojů a jejich disků bez nutnosti manuální editace konfiguračního souboru `qemu-backuper.json`.

Prvním parametrem je ve všech popisovaných příkladech název virtuálního stroje, který musí být shodný se zvoleným názvem v rámci konfiguračního souboru. Na základě tohoto parametru je při inicializaci nástroje vždy načtena konfigurace pro zvolený virtuální stroj. Pro demonstrační účely bude v rámci příručky používán název `debian10`.

C.1 Vytvoření plné a inkrementální zálohy

Pro vytvoření či plné inkrementální zálohy jsou určeny příkazy `full` a `inc`. Jediným povinným parametrem je `--backup-path-prefix`, který definuje cestu ke složce, kam má být výsledná záloha uložena.

Nepovinným parametrem je `--mode`, který definuje jakým způsobem má být požadovaná záloha vytvořena. Výchozí hodnotou je `suspend`, a v tomto případě dojde k pozastavení virtuálního stroje před získáním obsahu dirty bitmapy a vytvořením snímku zálohovaného disku, čímž dojde ke zlepšení konzistence zálohovaných dat za cenu krátké nedostupnosti virtuálního stroje. Druhou hodnotou je `unsafe`, při jejímž zvolení nedojde k pozastavení virtuálního stroje, a jeho dostupnost nebude ovlivněna. Při této metodě existuje nízké riziko vzniku nekonzistentní zálohy (viz 5.4.4). Příklad použití příkazu je uveden v ukázce kódu 5.3.

C.2 Obnovení dat ze zálohy

Pro obnovení dat ze zálohy slouží příkaz `build-img`, pomocí kterého je možné sestavit obraz disku do libovolného blokového zařízení nebo souboru. Pro tento příkaz jsou definovány následující čtyři povinné parametry:

- `--backup-path-prefix` - Cesta ke složce složce se zálohami (shodná jako v C.1).
- `--drive` - Absolutní cesta k disku, jehož záloha bude obnovována (shodná jako `filename` v konfiguračním souboru, viz 5.4.3).
- `--datetime` - Čas ve formátu ISO 8601 s přesností na sekundy, na základě kterého bude vybrána. použitá záloha pro sestavení obrazu disku. Pokud neexistuje záloha ve zvoleném čase, tak je vybrána nejbližší starší.
- `--output` - Cesta, do které bude uložen výsledný obraz disku (může být blokové zařízení i soubor).

Jediným nepovinným parametrem je `--restore` s prázdnou hodnotou, který by měl být použit v případě, že je prováděna obnova do původního blokového zařízení, ze kterého byla vytvořena záloha. Na základě tohoto parametru bude při další záloze disku vynucena plná záloha, jelikož po obnově disku není možné jednoduše navázat na stávající sekvenci inkrementálních záloh. Příklad použití příkazu je uveden v ukázce kódu 5.4.

C.3 Smazání zálohy

Ke smazání nepotřebné zálohy slouží příkaz `delete`, který zajistí smazání vybrané zálohy, a zároveň všech záloh, které jsou na této záloze závislé. Pokud je tedy spuštěn příkaz pro smazání plné zálohy, tak jsou smazány i všechny inkrementální zálohy, které byly vytvořeny vůči zvolené plné záloze. Povinnými parametry příkazu jsou `--backup-path-prefix`, `--drive` a `--datetime`, přičemž jejich význam je shodný jako v případě příkazu `build-img` (viz C.2).

```
qemu-backuper debian10 delete
    --backup-path-prefix /mnt/backups
    --drive /dev/drbd/by-res/vm-100-disk-1/0
    --datetime 2021-04-27T15:45:47
```

Ukázka kódu C.1: Příkaz pro smazání vybrané zálohy

C.4 Konfigurace nástroje

Pro jednodušší možnosti automatizace konfigurace nástroje byl implementován příkaz `config`, který umožňuje přidávání nových virtuálních strojů a jejich disků do konfiguračního souboru `qemu-backuper.json`.

C.4.1 Přidání nového virtuálního stroje

K přidání nového virtuálního stroje je určen podpříkaz `add-vm`, který přijímá jediný povinný parametr `--monitor`, pomocí kterého je předávána cesta ke QEMU monitoru přidávaného virtuálního stroje. Jako název virtuálního stroje je použita hodnota z prvního předaného argumentu.

```
qemu-backuper debian10 config add-vm --monitor /run/qemu-
    ↪ server/100.qmp


# Výsledný konfigurační soubor:
{
  "vms": {
    "debian10": {
      "monitor": "/run/qemu-server/100.qmp",
      "platform": null
    }
  }
}
```

Ukázka kódu C.2: Přidání nového virtuálního stroje do konfiguračního souboru

C.4.2 Přidání nového virtuálního disku

Pro přidání nového virtuálního disku pro zvolený virtuální stroj je určen podpříkaz `add-drive`, který přijímá následující parametry:

- `--type` - Typ přidávaného disku, na základě kterého bude zvolena metoda vytváření snímku. Podporovanými hodnotami jsou `lvm` a `linstor`.
- `--identifier` - Identifikátor, který slouží k nalezení disku při získávání konfigurace. všech disků z QEMU. Může obsahovat buď hodnotu `node-name` z konfigurace QEMU nebo shodnou hodnotu jako u parametru `--filename`.
- `--filename` - Absolutní cesta k blokovému zařízení disku.
- `--dirty-bitmap-path` - Absolutní cesta k souboru s dirty bitmapou, která bude využita pro účely inkrementálních záloh.
- `--dirty-bitmap-granularity` - Granularita dirty bitmapy, která je ve výchozím stavu nastavena na hodnotu 65536.

- Parametry pro typ disku `lvm`:
  - `--via-lvm-snapshot` - Cesta k LVM snímku, který bude vytvořen před zahájením procesu zálohy. Tato hodnota bude předána parametru `-n` nástroje `lvcreate`.
  - `--lvm-snapshot-size` - Velikost vytvářeného snímku, která bude předána parametru `-L`. nástroje `lvcreate`.
- Parametry pro typ disku `lvm`:
  - `--linstor-snapshot-name` - Název vytvářeného snímku, který bude vytvořen před zahájením procesu. zálohy.

```
qemu-backuper debian10 config add-drive
    --type lvm
    --identifier /dev/vg_storage/vm_100
    --filename /dev/vg_storage/vm_100
    --dirty-bitmap-path /bitmaps/vm_100.bitmap
    --dirty-bitmap-granularity 65536
    --via-lvm-snapshot /dev/vg_storage/vm_100_snap
    --lvm-snapshot-size 50G


# Výsledný konfigurační soubor:
{
  "vms": {
    "debian10": {
      "monitor": "/run/qemu-server/100.qmp",
      "platform": null,
      "drives": [
        {
          "type": "lvm",
          "identifier": "/dev/vg_storage/vm_100",
          "filename": "/dev/vg_storage/vm_100",
          "dirty_bitmap_path": "/bitmaps/vm_100.bitmap",
          "dirty_bitmap_granularity,": 65536,
          "via_lvm_snapshot": "/dev/vg_storage/vm_100_snap",
          "lvm_snapshot_size": "50G"
        }
      ]
    }
  }
}
```

Ukázka kódu C.3: Přidání nového virtuálního disku do konfiguračního souboru

V případě, že všechny zálohované disky využívají disky typu `linstor`, tak je možné provést jejich automatickou konfiguraci pomocí podpříkazu `qemu-sync-drives`, který získá konfiguraci všech virtuálních disků pomocí spuštěného virtuálního stroje, a následně ji uloží do konfiguračního souboru.