**UNIVERZITA KOMENSKÉHO V BRATISLAVE**

**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**Zálohovanie súborového systému do obsahom adresovaného úložiska**

**Bratislava 2015  
Michal Molec**

**UNIVERZITA KOMENSKÉHO V BRATISLAVE**

**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY**

**Zálohovanie súborového systému do obsahom adresovaného úložiska**

**Diplomová práca**

Študijný program: Aplikovaná informatika   
Študijný odbor: 2511 Aplikovaná informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Školiteľ: Mgr. Michal Molec, PhD.

**Bratislava 2015  
Michal Molec**

Naskenované zadanie záverečnej práce

**Čestné vyhlásenie**

Čestne vyhlasujem, že som diplomovú prácu "Zálohovanie súborové systému do obsahom adresovaného úložiska" vypracoval samostatne s použitím uvedenej literatúry a zdrojov dostupných na internete.

V Bratislave dňa 1.6.2015

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Michal Molec

**Poďakovanie**

Chcel by som sa poďakovať vedúcemu diplomovej práce Mgr. Jánovi Kľukovi, PhD. za cenné rady a usmernenie pri vypracovaní tejto práce.

**Abstrakt**

MOLEC, Michal: Zálohovanie súborového systému do obsahom adresovaného úložiska - Univerzita Komenského v Bratislave. Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej informatiky. Vedúci diplomovej práce: Mgr. Ján Kľuka, PhD. Bratislava, 2015.

Cieľom diplomovej práce je navrhnúť vhodné obsahom adresované úložisko dát (content-addressable storage) a implemetovať nad ním zálohovanie súborového systému unixového OS (napr. Linux), údržbu a prezeranie záloh a obnovu dát. Návrh samotného úložiska môže vychádzať z už existujúceho (napr. git), treba ho však upraviť s ohľadom na požadovaný účel. Očakáva sa kreatívne využitie (a prípadná úprava) vhodne zvolených techník kompresie (klasická bezstratová, rozdielová) a garbage collection, technológie FUSE, prípadne sledovania zmien v súborovom systéme. Implementácia by mala byť objektová, modulárna a rigorózne otestovaná (unit testing pokrývajúci funkcionalitu kritickú pre spoľahlivosť zálohovania). Preferovaným implementačným jazykom je python..

**Kľúčové slová:** Zálohovanie, FUSE, Rsync, Linux, Python

**Abstract**

MOLEC, Michal: File-system backup into content-addressable storage – Comenius University in Bratislava. Faculty of mathematics, physics and informatics, Department of aplied informatics. Dissertation supervisor: Mgr. Ján Kľuka, PhD. Bratislava, 2015.

The goal of this work is to design suitable content-addressable storage (CAS) and implement backup, backup maintenance and browsing, and data recovery of a Unix file system over the CAS. The design of the CAS may be based on existing ones (e.g., Git), it should, however, be adjusted for backup purposes. The student is expected to creatively choose, use, and adjust suitable methods of compression (standard lossless compression, delta compression) and garbage collection, the FUSE technology, and, possibly, the file-system event notification mechanism (inotify). The implementation should be object-oriented, modular, and rigorously tested (unit testing covering components critical to reliability of backup). The preferred implementation language is python.

**Keywords:** Backup, FUSE, Rsync, Linux, Python

**Obsah**

[Úvod 10](#_Toc405922246)

[1 Úvod do problematiky 11](#_Toc405922247)

[1 Návrh riešenia 11](#_Toc405922248)

[2 Implementácia 12](#_Toc405922249)

[Záver 10](#_Toc405922250)

[Bibliografia 11](#_Toc405922251)

[Prílohy 12](#_Toc405922252)

# Úvod

# Existujúce riešenia

V tejto kapitole budú uvedené existujúce riešenia zálohovania dát a ich základný princíp fungovania.

## Software

### Rdiff backup – duplicity

Rdiff backup je zálohovací software pre plne automatické inkrementálne zálohovanie a spätné zrekonštruovanie zálohovaných dát. Obnovanie dát môže prebiehať na konkrétne zvolenom súbore alebo celej zálohe. Implementovaný je v jazyku Python a pri svojom behu využíva program rsync.

Základné vlastnosti:

* Kopírovanie súborov lokálne alebo po sieti cez protokol SSH
* Neprenáša celé súbory ale len zmeny voči pôvodným (diff). Tieto zmeny sú ukladané.
* Zachovanie práv súborov a médií (zálohovanie na flash disk so súborovým systémom FAT)
* Obnovenie zmazaných súborov
* Štatistiky a záznamy o zálohách

### Amanda (Advanced Maryland Automatic Network Disk Archiver)

Amanda je zálohovací software, ktorý umožňuje zálohovanie dát po sieti LAN. Funguje na princípe jedného zálohovacieho servera, ktorý sa stará o zálohovanie viacerých klientov (Windows, Linux, Unix, BSD, Mac OS-X). Používa k tomu natívne dump a GNU tar nástroje.

Základné vlastnosti:

* Inkrementálne aj úplne zálohovanie
* Podpora šifrovania dát
* Zálohovanie viacerých platforiem – využíva k tomu buď samba protokol alebo svoj vlastný, natívny klient s názvom amanda-client
* Open Source
* Dáta su obnoviteľné aj bez samotného softwaru Amanda

### Time Machine

Time Machine je vstavaná zálohovacia funkcia systému Mac OS-X, ktorá zálohuje na externý disk alebo na takzvanú Time Capsule (Apple).

Základné vlastnosti:

* Podpora šifrovania dát
* Záloha celého systému, vrátane systémových súborov, aplikácií, účtov...
* Kompletný obraz systému pre daný čas – možná obnova
* Nezmenené súbory ukladá ako odkazy k pôvodným súborom a ukladá len nové, prípadne zmenené súbory
* Vytvárané sú hodinové zálohy za posledných 24 hodín, každý deň ak je disk pripojený
* Možnosť vyhľadávania súborov a rýchly náhľad do súboru

### Git

Git je decentralizovaný / distribuovaný systém riadenia revízií (DVCS). Neslúži však ako zálohovací software, ale ako software pre správu verzií. Bol vyvinutý pre vývoj Linuxového jadra. Väčšinou je však využívaný vývojármi pri práci na softwarových projektoch, ale dá sa využívať na akékoľvek dáta. Dá sa tak jednoducho zistiť, kedy sa daný súbor zmenil, ako sa zmenil a taktiež kým bol zmenený.

Základné vlastnosti:

* Git je distribuovaný systém, čo znamená, že nepotrebuje žiadny centrálny bod, s ktorým by musel udržiavať spojenie. Všetky základné operácie sú vykonávané lokálne.
* Kompletná história bez možnosti zmeny
* Kompatibilita s existujúcimi systémami a protokolmi: HTTP, FTP, SSH, rsync, alebo komunikácia priamo cez socket
* Garbage kolekcia
* Delta – kompresia
* Viaceré vetvy

## Techniky

### Obsahom adresované úložisko (content addressed storage)

Obsahom adresované úložisko dát funguje na nasledovnom princípe. Na strane klienta, je buď aplikačným serverom alebo samotným klientom vytvorený objekt na základe jeho obsahu (content address). Ten je následne odoslaný do úložiska a je pre neho vypočítaná adresa (hash), podľa ktorého je neskôr daný objekt jednoznačne identifikovateľný. Vypočítaná adresa je následne odoslaná späť aplikácií, ktorá pre ďalšie referencie na daný objekt, pracuje už len s ňou.

Výhody a nevýhody:

* Najvhodnejšie na dátach, ktoré sa často nemenia (kvôli častému počítaniu / prepočítavaniu adries pre jednotlivé súbory)
* Rýchle vyhľadávanie pre obsah daného súboru
* Nikdy neexistuje viac ako jedna kópia daného súboru v úložisku (dva také isté súbory majú rovnakú adresu – content address)

### Rsync

Rsync algoritmus bol vyvinutý Andrewem Tridgellom za účelom zeefektívnenia prenosu dát za podmienky, že sa na cieľovom počítači nachádza niejaká predchádzajúca verzia daných dát.

Algoritmus efektívne počíta rozdiely medzi súbormi pomocou kontrolných súčtov („checksumov“). V algoritme sú použité dva typy kontrolných súčtov („signatúr“), silný („strong checksum“) a slabý („weak checksum“ alebo „rolling checksum“) kontrolný súčet. Slabý kontrolný súčet musí byť veľmi rýchly a lacný na výpočet. Silný kontrolný súčet musí mať veľmi nízku pravdepodobnosť vzniku kolízie a je počítaný len vtedy, keď sa dva slabé kontrolné súčty zhodujú. K jeho výpočtu je používaný hashovací algoritmus MD5. Takto vypočítané kontrolné súčty sa posielajú medzi dvoma komunikujúcimi stranami a zisťujú sa tak rozdiely medzi nimi.

### FUSE

FUSE (Filesystem in Userspace) alebo tiež súborový systém v užívateľskom priestore umožňuje bežným uživateľom (Linux, Unix, FreeBSD, NetBSD, OpenSolaris, Mac OS-X) pristupovať k virtuálnym súborom systému. Taktiež umožňuje vytváranie súborových systémov z akýkoľvek dát a poskytovaných služieb.

Ovládač súborového systému je kernelový modul a je pevnou súčasťou samotného kernelu. Daný súborový systém môže obsluhovať len root a ten prideľuje práva k jednotlivým mountovacím bodom. Samotné použitie súborového systému funguje tak, že kernel dostane požiadavku od aplikácie, napríklad „vypíš adresár“, a ten prenechá ovládaču súborového systému. Ten následne požiadavku obslúži a spracuje odpoveď, ktorú kernel vráti späť aplikácii.

Samotný FUSE sa v kernely taktiež tvári ako obyčajný súborový systém. Nezaujíma sa však o obslúženie požiadavky a spracovanie odpovedi a namiesto toho, pomocou knižnice libfuse spustí bežnú používateľskú aplikáciu a tá vykoná všetko potrebné. Akonáhle aplikácia vráti odpoveď v požadovanej forme, FUSE ju prenechá kernelu a kernel ju vráti späť samotnej aplikácii.



Výhody:

* Celá logika súborového systému je zapísaná v programe prípadne skripte, ktorý je nezávislý na platforme či operačnom systéme
* Súborové systémy môže obsluhovať aj bežný uživateľ
* Stabilné API pre FUSE – nie je potrebné prepisovať programy

Existujúce FUSE aplikácie: DropboxFS, WikipediaFS, GmailFS

### Reference counting garbage collection

Najdôležitejší problém, ktorý riešia všetky garbage collectory, je priradenie objektov do dvoch skupín. Na objekty, ktoré sú dosiahnuteľné a na objekty, ktoré nie sú dosiahnuteľné.

Najjednoduchšou implementáciou garbage collectora je pomocou metódy Reference Counting. Funguje na nasledovnom princípe. Ku každému objektu je asiciovaný počet referencií na daný objekt. Ak hodnota klesne na 0, objekt je následne označený ako nedosiahnuteľný a je alebo bude uvoľnený z pamäte. Táto metóda však nedokáže pracovať s cyklickými štruktúrami.

# Návrh riešenia

V tejto kapitole je popísaný návrh riešenia našej aplikácie.

# Implementácia

V tejto kapitole bude rozoberaná konkrétna implementácia navrhnutých riešení z predchádzajúcej kapitoly a pre lepšiu predstavivosť a pochopenie budú uvedené aj konkrétne príklady.

# Záver

# Bibliografia

[1] ANDREW, Tridgell. Efficient Algorithms for Sorting and Synchronization [Dátum Júl 1999] http://gan.anu.edu.au/~brent/pd/Tridgell-thesis.pdf

[2] SCOTT, Chacon. Pro Git [Dátum 8. Február 2010] http://labs.kernelconcepts.de/downloads/books/Pro%20Git%20-%20Scott%20Chacon.pdf

# Prílohy

1. CD so zdrojovými kódmi aplikácie a bakalárskou prácou vo formáte PDF