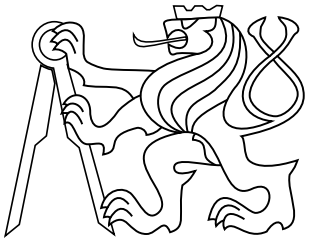
diplomová práce

Distribuované úložiště dat - správa dat

Bc. Martin Kudrnáč



Květen 2014

Vedoucí práce: Ing. Peter Macejko

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra počítačů

Poděkování

Chtěl bych vyjádřit poděkování panu Ing. Peterovi Macejkovi za jeho ochotu, vstřícnost, poradenství a připomínky v průběhu práce. Mé další poděkování musí směřovat mé rodině, jejíž členové plně chápali důležitost mého vytížení v posledním semestru a poskytovali mi dostatek volného času. Velké poděkování také patří všem ostatním studentům, kteří se mnou studovali a dělili se o své vědomosti. V poslední řadě nesmím zapomenout poděkovat všem vyučujícím, se kterými jsem se během svého studia setkal, za jejich vstřícnost a za veliké množství vědomostí, které mi umožnili poznat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Abstrakt

Cílem této práce bylo navrhnout a implementovat pilotní subsystém distribuovaného úložiště, který se stará o samotné uložení dat s zohledněním na výpadky a škálovatelnost daného řešení. Výsledkem je funkční program, který implementuje základní myšlenky a ukazuje koncepci dalšího rozvoje.

Klíčová slova

Distribuované úložiště; DHT; Datové centrum;

Abstrakt

The aim of this work is to design and implement a pilot subsystem of distributed storage that takes care of itself save the data with consideration to outages and scalability the solution. The result is a functional program, which implements the basic idea that shows a concept for further development.

Keywords

Distributed storage; DHT; Data center;

Obsah

Seznam obrázků

Seznam tabulek

Zkratky

AES	Advanced Encryption Standard
AMD	Advanced Micro Devices
API	Application programming interface
AS	Access Server
CPU	Central processing unit
DDR	Double data rate
DHT	Distributed Hash Table
HDD	Hard disk drive
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet protokol
NAT	Network address translation
PGP	Pretty Good Privacy
POSIX	Portable Operating System Interface
P2P	Peer-to-peer
RAM	Random-access memory
RAID	Redundant Array of Inexpensive/Independent Disks
REST	Representational State Transfer
SAS	Serial Attached Small Computer System Interface
SFU	Services for UNIX
SHA	Secure Hash Algorithm
SMP	Symmetric multiprocessing
SQL	Structured Query Language
SSL	Secure Sockets Layer
SUA	Subsystem for UNIX-based Applications
TCP	Transmission Control Protocol
TLS	Transport Layer Security
VCPU	Virtual central processing unit

1. Úvod

V minulé, dnešní i budoucí době bylo, je a bude ukládání datových informací velice důležitým faktorem. Zvláště v dnešní elektronické době je způsob ukládání elektronických dat klíčovým faktorem při práci s moderními systémy. V minulosti se ukládalo pouze lokálními způsoby. Jako například na děrné štítky, magnetické pásky a až po klasické pevné disky.

Vývoj těchto uložiť je stále v pohybu. Dnes už má každý uživatel možnost za přijatelné peníze vlastnit úložiště s kapacitou tera bajtů dat. Ovšem s vývojem moderních technologii přichází další důležitý faktor, který komplikuje tuto koncepci. Problém nastává pokud chce k datům přistupovat odkudkoliv. Řešeních která tento problém řeší je hned několik. Od jednoduchého nahrávání dat na přenosný disk až po vytvoření datového serveru, ke kterému budeme přes internet přistupovat.

Problém, ale nastává v případě kdy je pro nás důležitá dostupnost. Ano, máme data přístupná odkudkoliv, ale je vše závislé na funkčnosti jednoho zařízení. Problémy těchto koncepcí lze částečně vyřešit decentralizací celého systému. Nevýhodou těchto systémů je ovšem mnohonásobně komplikovanější vývoj, správa a údržba. Výhodami jsou naopak značné navýšení propustnosti, vyšší dostupnost a celková odolnost proti výpadkům.

Na současném trhu je mnoho služeb co nabízejí ukládání dat do distribuovaného úložiště. Nicméně každá z nich se soustřeďuje na různé typy zákazníků v otázce bezpečnosti, spolehlivosti, rychlosti a ceny.

Tato práce se tedy snaží společně s prácemi Bc. Jana Janury a Bc. Tomáše Dyntara vytvořit koncepci systému, který by všechny tyto otázky snažil zahrnout. Konkrétně cílem této práce je navrhnout subsystém, který se bude starat o samotné uložení dat.

2. Popis problému specifikace cílů

2.1. Specifikace cíle

Cílem této práce je se seznámit s dostupnými cloudovými úložišti. Poté s přihlédnutím k poznatkům vypozorovaných během toho seznámení navrhnout a vytvořit pilotní verzi subsystému distribuovaného úložiště, které se stará o samotné uložení dat. Tento subsystém bude navrhnut s přihlédnutím k nasazení do firemního prostředí. Proto je nutné se zaměřit na bezpečnost a spolehlivost.

2.1.1. Struktura práce

V následující části si představíme strukturu této práce. V první části si vysvětlíme co distribuované úložiště dat je a jaké má výhody a nevýhody.

V druhé části se podíváme na stávající služby na trhu, která poskytují cloudové úložiště. Popíšeme si je z hlediska sdílení, synchronizace, bezpečnosti a jejich klientských aplikací. Nakonec každé řešení otestujeme. V závěru poté porovnáme všechny zmíněné služby a zobrazíme získané informace do přehledové tabulky. Tato tabulka bude rovněž obsahovat informace o dalších úložištích od kolegů Bc. Jana Janury [1] a Bc. Tomáše Dyntara. [2]

V třetí části provede lehký úvod do celkového projektu distribuovaného úložistě.

V další části se zaměříme na návrh subsystému, který se stará o ukládání dat v rámci projektu distribuovaného úložiště. Dále zanalyzuje základní možné výpadky v rámci celého systému spolu s možnými řešeními jak tyto výpadky ošetřit.

Následující kapitola realizace se bude věnovat pilotní implementaci subsystému z analýzy. Zaměříme se zde na popis jednotlivých modulů a komunikací v rámci celkového systému.

Kapitola testování se bude zabývat testy na realizované pilotní implementaci. Budeme testovat funkčnost a propustnost navrženého systému.

V závěru si zhodnotíme cíle práce a co se nám podařilo splnit a navrhneme, jak se dá práce dále rozšiřovat.

Nakonec v příloze nalezneme uživatelskou příručku k pilotní implementaci – informaci jak implementaci nainstalovat, nastavit a spustit.

2.2. Základní pojmy a principy fungování

2.2.1. Distribuovaný systém

Distribuovaný systém je takový výpočetní systém, který zahrnuje vice než jednu výpočetní jednotku (procesor/počítač). Vlastní program, který je rozdělen na části, které si vzájemně předávají data. Tyto data jsou pak zpracovány na různých spolupracujících jednotkách. Tedy nesdílejí společnou paměť. [3]

Takovýto systém se uživateli jeví jako jednotný systém. Uživatel tedy komunikuje se systémem jako celkem a nestará se o topologii, počty uzlů, nebo komunikaci.

2.2.2. Distribuované uložiště

Distribuované úložiště je tedy podle definice distribuovaného systému, systém, který se nachází na vícero počítačích a uživateli se jeví jako jeden celek. Tudíž si toto úložiště můžeme představit jako například síťový disk. Potencionální uživatel tedy může s úložištěm pohodlně pracovat jako například s klasickým sambovým diskem. Pro základní operace čtení a zápis je použití stejné. Nicméně distribuované úložiště může jednotlivá data ukládat na více různých počítačů. Data také mohou být přesouvána na jiné počítače v rámci systému aniž by uživatel něco zpozoroval.

Výhody

- Rozšiřitelnost - Jednotlivé servery je možné za běhu přidávat, nebo naopak odebírat.
- Výkonnost - Díky redundanci lze zařídit rozložení zátěže mezi jednotlivé servery.
- Spolehlivost - Jednotlivá data jsou rozložena na více různých serverů a právě díky tomu, výpadek serveru nevede nutně k ztrátě dat.
- Cena - Je možné použít velký počet cenově dostupných serverů a spojit je do jednoho celku.
- Přizpůsobivost - každé jednotlivé datové centrum je schopno fungovat samostatně. Jednotlivá data mohou být přemístěny na jiná datová centra.

Nevýhody

- Nedostupnost dat - data nemusí být uživateli dostupná pokud není připojen k síti. Nicméně tento nedostatek lze vyřešit uživatelskou vyrovnávací pamětí.
- Bezpečnost - Snadnější přístup k datům, jako například proudění dat přes veřejnou síť.
- Větší komunikace - díky velkému počtu serverů jsou vyšší nároky na síť. Může se stát, že komunikace zahltí síť.

3. Rešerše stávajících řešení

V následující rešerši jsem analyzoval některá existující řešení datových úložišť. Jednotlivá řešení jsem porovnával z hlediska sdílení, synchronizace, bezpečnosti a jejich klientských aplikací. Na konci je souhrnné porovnání všech nejvýznamnějších datových úložišť ve formě přehledné tabulky, která je zobrazena na Obr. 3. Tato souhrnná tabulka vznikl spojením spolu rešeršemi kolegů Bc. Jana Janury [1] a Bc. Tomáše Dyntara [2]. tato rešerše vznikla v únoru roku 2014.

3.1. Testovací sestavy

3.1.1. Testovací data

Testovací data byla trojího typu, tak aby byla prověřena rychlost napříč velikostí souborů. První typem byl soubor o celkové velikosti 1495MB. Jednalo se o ISO obraz disku. Druhým typem bylo 152 fotografií o celkové velikosti 501MB. Posledním typem bylo 15761 souborů o celkové velikosti 87.8MB, které byly použity z části jádra operačního systému Linux.

3.1.2. Osobní počítač

Testování probíhalo na osobním notebooku na kterém běžel ve virtuálním prostředí Oracle VirtualBox (verze 4.3.6) s operačním systémem Microsoft Windows 8 64bit. Virtuálni stroj disponoval dvěmi virtuálními procesory a operační pamětí 2GB. Samotný notebook obsahoval procesor Intel Core2Duo P8400 s operační pamětí 4GB. Notebook byl připojen do kolejní sítě Masarykovy koleje o konektivitě 100Mbit/s. Pro testování byl vytvořen obraz systému, který byl obnovován vždy po otestování služby. V testech středních souborů bylo testováno 100 fotografií.

3.1.3. Referenční sestava

K relevantnímu testování služeb byly testy prováděny na virtuálním stroji v počítačové síti ČVUT FEL. Hlavním důvodem byla relevantnost v porovnání s ostatními službami, které testovali kolegové. Samotný virtuální stroj obsahoval systém Microsoft Windows 7 64bit a byla vytvořen snapshot čistého systému. Po každém otestování služby byl systém obnoven zmíněným snapshotem.

Parametry sestavy na kterém běžel virtuální stroj jsou následující:

- Hardware - Dell PowerEdge R710 [4] s 2x Intel Xeon X5667 a diskovým polem Clariion AX4 [5]
- Software - Scientific Linux 6.3

Pro zaznamenávání datového toku byl použit program Wireshark. Pro měření zatížení procesoru byl použit integrovaný nástroj Sledování systému (Perfmon) systému Microsoft Windows 7.

Testované soubory byly stejné jako při testování na osobním počítači. Jediná změna

byla ve velikosti středních souborů, kdy byl navýšen počet fotografií na 152. Důvodem byla vyšší směrodatnost k přenosové rychlosti.

3.2. Microsoft SkyDrive

Aplikace vyvinutá společností Microsoft a vypuštěna prvního srpna roku 2007. Dříve byla publikována pod názvy Microsoft Live Folders respektive Microsoft Live SkyDrive. Jedná se o klasické cloudové úložiště se synchronizací s dalšími doplňkovými službami.

Samotný název úložiště projde v blízké době k přejmenování. Jednou z možností je název OneDrive, který by měl být i použit. Bohužel pro Microsoft se k tomuto negativně vyjádřila společnost One, která poskytuje cloudové služby. Změna názvu SkyDrive je z důvodu rozhodnutí soudu, který zkoumal stížnost britské televizní společnosti British Sky Broadcasting. Podle stanice Sky je název kopírováním jejich názvu a matoucí pro jejich zákazníky. Něco podobného už Microsoft musel řešit v nejnovějším operačním systému Windows 8, kdy nové uživatelské prostředí označoval jako Metro. Nicméně, aby Microsoft předešel soudnímu sporu s německou společností Metro AG, změnil název na Modern UI.

Metodami přístupů jsou webové rozhraní, lokální klient a mobilní aplikace. Podporované platformy jsou Windows, Mac OS, iOS, Windows Phone a Android. Nedostatkem je ne podpora platformy Linux. Pro přístup a využívání služby je uživatel povinen mít vytvořený Microsoft účet.

Zdarma nabízená velikost pro uživatele je 7GB, což je v porovnání s konkurencí vyšší standart. Rozdíl mezi placenou a bezplatnou verzí je pouze ve velikosti úložného prostoru. Pro placené navýšení velikosti jsou ceny zobrazeny v Tab. 1. Z Tab. 1 je tedy vidět, že služba SkyDrive patří mezi ty nejlevnější na trhu. Samozřejmostí je i individuální dohoda podle zákaznických požadavků.

Tarif	Cena [Kč/rok]	Úložný prostor [GB]
20	190	27
50	480	57
100	960	107
200	1920	207

Tabulka 1. Přehled tarifů služby SkyDrive

Omezení na velikost ukládaných souborů je stanovena na 2GB. Přes webové rozhraní je maximální dovolená nahrávací velikost 300MB. Data se přenášejí přes zabezpečený kanál pomocí SSL.

Služba je svázaná s dalšími službami Microsoftu, jako je poštovní klient Outlook, lidé a kalendář.

Kladem aplikace je velice široká jazyková podpora včetně češtiny

3.2.1. Sdílení a synchronizace

Jednotlivé soubory uložené na datovém úložišti mohou být následně sdíleny. Možnosti sdílení doznali změny k lepšímu oproti minulosti. Nyní je možné sdílet soubory po skupinách. To tedy například znamená sdílet své fotografie z různých složek jen s určitým datem. Sdílení je umožněno prostřednictvím informativního mailu, který obsahuje v tomto případě náhledy fotografií. Ke každému sdílenému souboru lze nastavit oprávnění užívání. Další funkcí je možnost upravovat dokument emailem bez přihlášení, pokud uživatel vlastní oprávnění.

Další funkcionalitou je možnost zobrazení činnosti uživatele (co dříve sdíleli, kdy a s kým).

Možnosti synchronizace jsou zde značně omezené. Uživatel má možnost synchronizovat pouze soubory a složky umístěné v složce „SkyDrive“. Nelze zde jakkoli si nechat zobrazit aktuálně zpracovanou frontu, nebo nastavit si maximální rychlost pro nahrávání.

3.2.2. Bezpečnost

Bezpečnost není na vysoké úrovni jako například u konkurenční Wualy. Chybí zde jakékoliv šifrování na straně klienta či serveru. Další relativní bezpečnostní chybou je, že aplikace po naistalování se už nikdy neptá na přihlašovací údaje jak u mobilní aplikace tak u lokální aplikace. Tedy po zcizení například telefonu jsou soubory uložené na internetu zcela k dispozici.

Největší nevýhodou této služby z mého hlediska je filtrování nevhodného obsahu na úložišti. Toto opatření se vztahuje na veškerá uložená data, tedy i soukromá. Pokud uživatel nahraje jakkoliv nevhodný obsah a to i v rámci textu, kdy se jedná o vulgarismy, nebo kresby či fotografie pro dospělé bude jeho účet nekompromisně zrušen. Microsoft k filtrování využívá jeho vlastně vyvinutou technologii PhotoDNA [6], jejíž licenci používá i například Facebook. Veškerý obsah je tedy automaticky zkoumán a vyhodnocuje co je přijatelné a co ne. Je nutno podotknout, že algoritmus nefunguje bezchybně. Obsah nemusí nutně nesplňovat podmínky užívání, a přesto bude účet zrušen. Jedná se například o fotografie z pláže. Problém nastává u technologie PhotoDNA v tom, že se vyhodnocuje kolik procent „kůže“ je na obrázku.

3.2.3. Webový klient

Klient je velice přehledný a jednoduchý pro běžného uživatele. Předlohou pro grafickou realizaci bylo nové prostředí Windows UI známé ze systémů Windows 8.x. Služba nabízí podobnou funkcionalitu jako společnost Google u svého cloudového řešení Google Drive a to zobrazení vyhledávaných souborů přímo ve výsledcích hledání prohlížeče Google respektive Bing u Skydrive . V případě operačního systému Windows 8.x se tato funkcionalita uskutečňuje přímo v integrovaném vyhledávači v prostředí Windows UI. Prostředí se jeví jako svižné a informovuje o různých prováděných akcích.

Velice zajímavou funkcionalitou je integrace Office Web Apps (Word, Excel, PowerPoint, OneNote). Za zmínku také stojí možnost automatického ukládaní souboru Office do formátů odt, odp a ods. Webové rozhraní dále umí slideshow a zobrazení geoznaček u fotografií, nebo označování lidi na fotografii.

Zajímavostí pro vývojáře je integrace webového editoru, který umí upravovat html, js nebo txt. Dále umí zvýrazňovat syntaxi pro některé jazyky s jednoduchým našeptávačem. Umí také zobrazit dávky pro příkazový řádek, nebo soubory registrů.

Specialitou je vzdálený přistup k počítači, který má nainstalovaného lokálního klienta. Nicméně od této funkce Microsoft nejspíš ustupuje z důvodu bezpečnostního rizika (ve Windows 8.1 už není podporována).

3.2.4. Lokální klient

Klient postrádá mnoha kvalit webového rozhraní. Aplikace umí synchronizovat pouze předem preferovanou složku. Tuto zápornou vlastnost lze minimalizovat doplňkem SkyShellEx [7], který přidává synchronizaci jakékoliv složky v počítači. Nicméně je stále více integrován do novějších operačních systémů, jako tomu je u operačního systému Windows 8.x., nebo Windows Phone 8.x. Nastavení tu není prakticky žádné. Aplikace umí alespoň zobrazit uživateli stav synchronizace v počtu zbývajících souborů.

Pokud uživatel vlastní kancelářský balík Office 2010 a novější lze synchronizovat data přímo s uložištěm.

Funkcionalita verzování je dostupná pouze pro dokumenty kancelářské sady Office nicméně se solidním počtem verzí (25). API aplikace je veřejné.

3.2.5. Testování rychlosti

Jelikož webové rozhranní se jeví jako velice rychlé, byl předpoklad rychlosti i v otázce přenosových rychlostí u stahování a nahrávání pomocí synchronizace. Jak měření ukázala v Tab. 2 a Tab. 3 v otázce malých souborů je toto úložiště pro rychlé synchronizace takřka nepoužitelné. Nicméně soubory se na úložiště nahrají a aplikace nejeví známky zamrznutí či spadnutí. Průměrná rychlost u velkého souboru a fotografií byla na podobně stejné úrovni. Celkové zatížení systému se pohybovalo mezi 5% až 20% jak je vidět na tabulce Tab. 4.

Počet souborů	Celková velikost souborů[MB]	Průměrná rychlost nahrávání [Mbit/s]	Celková doba nahrávání	Průměrná rychlost stahování[Mbit/s]	Celková doba stahování
1	1495	3.2	1h3m	5.54	36m
100	331	3	15m	6.3	7m
15761	87.8	0.7	3h3m	0.23	51m

Tabulka 2. Přehled testování rychlostí služby SkyDrive na osobním notebooku

3.3. SugarSync

Služba SugarSync byla vyvinuta v společností Sharpcast, která byla založená v roce 2004. V roce 2006 byla představena služba Sharpcast Photos. Služba, umožňovala syn-

Počet souborů	Celková velikost souborů[MB]	Průměrná rychlost nahrávání [Mbit/s]	Celková doba nahrávání	Průměrná rychlost stahování[Mbit/s]	Celková doba stahování
1	1495	4	58m	2.8	1h19m
152	501	2.83	21m31s	5.315	51m
15761	87.8	0.23	4h12m	0.621	1h15m

Tabulka 3. Přehled testování rychlostí služby SkyDrive na referenčním serveru

Počet souborů	Zatížení při nahrávání	Zatížení při stahovánií
1	15%	15%
152	10%	20%
15761	5%	10%

Tabulka 4. Přehled zatížení systému u služby SkyDrive na referenčním serveru

chronizaci obrázku mezi vícero zařízeními. V roce 2009 se společnost Sharpcast přejmenoval na SugarSync, Inc.. Tento počin znamenal přechod z názvu Sharpcast Photos na název SugarSync.

SugarSync je představitelem klasické cloudové služby pro široké masy uživatelů. Metodami přístupů jsou opět webový klient, desktopový klient a webová aplikace. Podporované operační systémy jsou Windows, Mac OS, iOS, Android, BlackBerry, Symbian. Operační systém Linux není podporován, nicméně tento nedostatek lze vyřešit pomocí programu Wine [8]. Jde tedy o velice široké spektrum podporovaných platforem.

Mezi nejvýznamnější uživatele služby patří například Lenovo, SanDisk, Korea Telecom, France Telecom-Orange. Společnost také přímo spolupracuje se společností Samsung. Díky této spolupráci jsou videa, fotky a hudba dostupná na zařízeních Samsung pomocí aplikace AllShare Play [9].

Pro bezplatné používání je k dispozici 5GB dat. Placené tarify jsou uvedeny v Tab. 5.

Tarif	Cena [$/rok]	Úložný prostor [GB]
Individual	74.99	60
Individual	99.99	100
Individual	249.99	250
Business	550	1000
Business	784-7962	1000+

Tabulka 5. Přehled tarifů služby SugarSync

Jak lze vypozorovat, placená služba patří k dražším na trhu. Tarify Bussiness přidávají další funkce jako například možnost vytvoření vícero uživatelů, nastavení admin účtů (nastavování kót pro uživatele, monitorování aktivity, nebo nastavení práv uživatelům), nebo online telefonická podpora . Samozřejmostí je i smluvní dohoda na tarif podle zákaznických podmínek.

Zajímavou vlastností je navýšení úložného prostoru pomocí získání nových uživatelů. Nicméně se jedná pouze o 500MB za každého nového uživatele. Další možností bezplatného navýšení služby je tak zvané „využití služby na plno“. Jedná se o navýšení 125MB za využívání jednotlivých funkcí. Jedná se například o využívání mobilního klienta, veřejné sdílení souboru či složky, privátní sdílení složky či souboru s přáteli.

Kladem služby je 30-denní možnost nezávazného bezplatného používání.

SugarSync nabízí velice nadstandardního mobilního klienta. Nadstandardem je například možnost streamování vlastní hudby uložené v cloudu. Dále je možné aplikaci zabezpečit pomocí PIN.

3.3.1. Bezpečnost

SugarSync API využívá REST architekturu přes HTTPS. Vlastní API je veřejné. Soubory jsou nahrávány přes zabezpečený kanálu pomocí TLS (SSL 3.3) a šifrovány na serveru pomocí 128bit kódování AES. SugarSync tedy vlastní klíče k zákaznickým datům. Žádosti a odpovědi využívají straightforward XML. Ukládáni dat se děje ve dvou separátních data centrech včetně Amazons S3 [10].

3.3.2. Synchronizace a sdílení

Synchronizace je na velice dobře úrovni. Samotné úložiště se na desktop počítači vytvoří jako lokální virtuální disk. Pro lepší práci s úložištěm lze nastavit vyrovnávací paměť v rozmezí 2 až 50GB. Dále je možné synchronizovat jakkoliv složku či soubor v počítači. Pro synchronizaci je integrace přímo v kontextovém menu operačního systému. Nevýhodou utility sloužící pro synchronizace je absence nastaveni rychlosti stahování. Odesílání dat má alespoň relativní nastavení rychlosti. Naopak kladnou vlastností je možnost nastavení priorit u nahrávaných souboru do cloudu.

Další vlastností je takzvaný „Magic Briefcase“ což je složka která automaticky synchronizuje veškeré obsažené soubory na cloud.

Zajímavou funkcionalitou, nicméně z mého hlediska zbytečnou, je dálkové mazání synchronizovaných dat ze ztraceného nebo odcizeného počítače. Tato služba je nicméně přístupná pouze platícím uživatelům.

Verzování je podporováno, nicméně počet verzí je stanoven na pět.

Sdílení souborů je na standardní úrovni. Uživatel může vytvářet veřejné odkazy. Je zde možnost sdílení pouze se skupinami lidí pomocí zaslané pozvánky na mail, nebo veřejného sdílení s přímým publikováním na sociálních sítích (Facebook, Twitter). Možnosti nastavení práv u sdílení jsou velice chudé. Buď vybraná skupina může soubory číst, nebo je plně ovládat.

3.3.3. Webový klient

Je nejslabší službou. Byl zde odebrán přehrávač medií, který činil webové rozhraní velice zajímavým. V otázce přehlednosti je na velmi dobré úrovni. Nicméně je zde malá jazyková podpora. Reakčnost klienta se ukázala jako velice pomalá.

Najdeme zde možnost zobrazení posledních aktivit prováděných na úložišti.

Oproti hlavním konkurentům chybí SugarSync integrace nějakého kancelářského balíku.

3.3.4. Lokální klient

Se jeví jako velice povedený. Samotný klient je integrovaný jako klasická aplikace s vlastnosti „drag and drop“ což značně uživateli zjednodušuje práci. V samotné aplikaci je zobrazen celkový obsažený prostor z poskytnutého a samotné složky na cloudovém úložišti. Jsou zde další dvě záložky na přepnutí, na možnosti sdílení a zobrazení aktivit, které byly na úložišti vykonány. Je zde obsažena i funkce na vyhledávání.

Dále aplikace namapuje virtuální složku přímo do systému spolu s integrací do kontextového menu.

3.3.5. Testování

Během testování se ukázala že i služba SugarSync není na práci s malými soubory. Služba měla problémy s nahráváním, kdy několikrát zamrzla. V otázce velkých a středních souborů už služba patří mezi ty nejlepší a nejevila jakékoliv známky nestability. Otestovat rychlost nahrávání pro velký soubor na testovém serveru se bohužel nepodařilo, z důvodu existence souboru od jiného uživatele v cloudu SugarSync. Údaje jsou zobrazeny v Tab. 6 a Tab. 7. V otázce zatížení systému se služba jeví jako náročnější, zvlášť při nahrávání, nebo stahování malých souborů dosahuje 20% až 25% zatížení. Celkové průměrné časy zatížení procesoru jsou uvedeny v Tab. 8.

Počet souborů	Celková velikost souborů[MB]	Průměrná rychlost nahrávání [Mbit/s]	Celková doba nahrávání	Průměrná rychlost stahování[Mbit/s]	Celková doba stahování
1	1495	64	3m	6.08	33m
100	331	3	15m	2.45	18m
15761	87.8	0.7	3h3m	0.23	51m

Tabulka 6. Přehled testování rychlostí služby SugarSync na osobním notebooku

Počet souborů	Celková velikost souborů[MB]	Průměrná rychlost nahrávání [Mbit/s]	Celková doba nahrávání	Průměrná rychlost stahování[Mbit/s]	Celková doba stahování
1	1495	-	-	2.9	1h20m
152	501	7.15	7m15s	2.67	30m41s
15761	87.8	0.4	1h50m	0.33	5h13m

Tabulka 7. Přehled testování rychlostí služby SugarSync na referenčním serveru

Počet souborů	Zatížení při nahrávání	Zatížení při stahovánií
1	40%	10%
152	15%	10%
15761	25%	20%

Tabulka 8. Přehled zatížení systému u služby SugarSync na referenčním serveru

3.4. Wuala

Služba, která je provozována renomovanou společností LaCie. Wuala byla vyvinuta švýcarským federálním institutem technologií (Swiss Federal Institute of Technology - ETH) se sídlem v Zurichu. Původně vyvinuta jako P2P síť.

Metodami přístupu jsou lokální klient, mobilní aplikace. Nevýhodou je absence webového rozhraní. Důvodem je zajištění bezpečnosti pro zákazníka [11].

Podporované platformy jsou Windows, Mac OS, iOS, Android, Linux. Platforma služby Wuala je postavená na jazyce JAVA, jako architekturu pro rozhranní používá REST. Jako jedna z mála služeb na trhu nenabízí veřejné API. Architektura Wualy je založena na distribuční hashovaní tabulce Chord, konkrétněji na Kangoo DHT.

Pro bezplatné používání je uživateli nabídnuta standardní velikost 5GB. Placené tarify jsou uvedeny v Tab. 9. Zajímavým parametrem je garantovaná dostupnost dat, která je stanovena na 99.99%. Maximální velikost souboru je stanovena na 40GB.

Tarif	Cena [Euro/rok]	Úložný prostor [GB]
Rozšíření FREE	29	20
Rozšíření FREE	65	50
Rozšíření FREE	109	100
Rozšíření FREE	219	200
Rozšíření FREE	549	500
Rozšíření FREE	999	1000
Rozšíření FREE	1799	2000
Business Groups	389	100

Tabulka 9. Přehled tarifů služby Wuala

Cena tarifů patří k těm dražším co lze na trhu nalézt. U služby je také možnost bezplatného navýšení úložného prostoru přes přivedení nového uživatele do služby. Tento počin je odměněn navýšením o 1GB. Tento mechanismus můžeme opakovat maximálně desetkrát.

Služba nabízí širokou jazykovou podporu včetně češtiny.

3.4.1. Bezpečnost

Úložiště Wuala si velice zakládá na bezpečnosti. Na současném trhu patří k těm nejlepším co se týče právě zabezpečení. Samotný princip zabezpečení je uplatňován tak, že na straně klienta se data před odesláním zašifrují pomocí šifry AES-128 a poté jsou

soubory rozděleny do několika bloků a následně jsou odesílány na různá místa. Pro výměnu šifrovacích klíčů je použita šifra RSA-2048. Data uživatele jsou navíc uchovávány redundantně po Evropě (Švýcarsko, Francie, Německo). Pro přenos používá SSL.

Díky vyšší míře zabezpečení zde nenajdeme webového klienta. Což pro normálního uživatele je nevýhoda. Společnost tento počin argumentuje jako nutný krok k zajištění vyšší bezpečnosti. Tudíž zde nenajdeme webové funkcionality jako u ostatních webových úložištích jako SkyDrive nebo Google Drive.

Další vlastností služby je neautomatické přihlášení do služby po startu z lokálního, nebo mobilního klienta. Tato vlastnost ačkoliv se zdá samozřejmá tak jí nenabízí mnoho konkurenčních služeb. Vždy je tedy uživatel nucen zadat přihlašovací jméno a heslo pro vstup na úložiště při spuštění služby. Lze, ale i nastavit aplikaci na automatické přihlašování.

Každý uživatel může vytvářet kontakty s ostatními uživateli a poté s nimi vytvářet skupiny

Speciálními funkcemi u souborů jsou například přidávání komentářů, nastavení omezení na 18+, přidání mezi oblíbené soubory.

3.4.2. Sdílení a synchronizace

Možnosti sdílení jsou trojí. První možností je sdílení přes webovou adresu. Druhou možností je sdílení s uživateli Wuala. Třetí možností je nastavit sdílení na veřejné, kdy je obsah viditelný všem uživatelům služby Wuala. Nevýhodou při sdílení je nemožnost sdílení jednotlivých souborů, ale nutnost sdílet celou složku.

Funkce verzování je zastoupena deseti zpětnými verzemi u všech souborů. Zobrazení verzí je zde řešeno velice elegantně pomocí „zobrazení cestování v čase“ kdy pomocí posuvníku se zobrazují jednotlivé změny. Dále je zde obsažen koš i možnost obnovy souborů, které byly smazány v koši.

Možnosti synchronizace jsou na nadstandardní úrovni. Můžeme synchronizovat jakoukoliv složku v počítači s možností vyloučení skrytých souborů, dočasných souborů, systémových soubory, nebo námi specifikovaných souborů.

Vlastní průběh synchronizace si může uživatel zobrazit a upravovat její průběh.

Klient dále nabízí i funkci zálohování spolu s nastavením, kdy se má dané zálohování pro konkrétní službu provádět. Je zde opět jako u synchronizace možnost pozastavení prováděné akce.

Zajímavou funkcionalitou při nahrávání fotek je možnost automatické komprese (změny rozměrů).

3.4.3. Lokální klient

Klientská aplikace je řešena velice komplexně. Vlastní úložný prostor se do systému namapuje jako síťový disk a dále se s ním i tak zachází.

Samotná aplikace se zobrazuje v přehledném okně, které je do značné míry podobné oknu z průzkumníka Windows. Dále se aplikace v systémech Windows integruje do kontextového menu.

Možností nastavení jsou nadstandardní. Klient má možnost přesně specifikovat šířku pásma pro nahrávání, nebo stahování. Je zde zobrazen aktuální stav sítě, rychlost nahrávání a stahování a indikátor zaplnění úložného prostoru. Také lze zde nastavit velikost vyrovnávací paměti programu. Je zde obsaženo nastavení informativních zpráv na události, jako například nové komentáře k uživatelským souborům, nebo na nové soubory uživatelovo známých osob.

Jednou z dalších funkcí je přístup z lokální sítě.

3.4.4. Testování

Během testování se služba Wuala ukázala, jako jedna z nejschopnějších služeb. Tato služba si velice schopně poradila s velkým počtem souborů, kdy nahrávání trvalo 30min. V otázce velkých a středních souborů se ukázala dokonce jako nejlepší služba mezi mnou testovanými. Veškeré údaje jsou zobrazeny v tabulce Tab. 10 a Tab. 11. V otázce zatížení systému byla tato služba jako nejnáročnější. Nicméně zatížení pramení z krátkých časů nahrávání tak stahování a šifrováním souborů. Údaje o zatížení jsou zobrazeny v Tab. 12.

Počet souborů	Celková velikost souborů[MB]	Průměrná rychlost nahrávání [Mbit/s]	Celková doba nahrávání	Průměrná rychlost stahování [Mbit/s]	Celková doba stahování
1	1495	28.5	7m	6.7	30m
100	331	4.2	10m	8.8	5m
15761	87.8	0.2	59m	1.3	9m

Tabulka 10. Přehled testování rychlostí služby Wuala na osobním notebooku

Počet souborů	Celková velikost souborů[MB]	Průměrná rychlost nahrávání [Mbit/s]	Celková doba nahrávání	Průměrná rychlost stahování [Mbit/s]	Celková doba stahování
1	1495	28.5	6m	7.3	30m41s
152	501	2.2	13m13s	10.8	6m58s
15761	87.8	0.7	30m	2	7m

Tabulka 11. Přehled testování rychlostí služby Wuala na referenčním serveru

Počet souborů	Zatížení při nahrávání	Zatížení při stahovánií
1	50%	20%
152	50%	35%
15761	20%	60%

Tabulka 12. Přehled zatížení systému u služby Wuala na referenčním serveru

3.5. SpikeOak

Služba SpikeOak patří společnosti SpikeOak, Inc., která byla založena v roce 2007. Společnost sídli v Northbrooku ve státě Illinois. V současné době má partnerství se společnostmi Qyen Digital LLC a CPP North America, LLC. Servery služby se nacházení výhradně na území USA.

Metodami přístupu jsou webový klient, mobilní aplikace a lokální klient.

V základní nabídce služba nabízí 2GB zdarma. V otázce omezení velikosti souborů zde nenajdeme žádný limit. Pro bezplatné navýšení kapacity lze využít možnosti přivedení dalšího uživatele. Tato činnost je odměněna navýšením o 1GB. Tuto možnost lze využít maximálně desetkrát. Placené tarify jsou rozděleny do dvou základních skupin, osobní a enterprise. Jejich ceny jsou zobrazeny v Tab. 13.

Tarify enterprise jak z názvu vypovídá slouží pro firemní nasazení. Základním rozdílem oproti osobnímu tarifu je možnost stovky uživatelů na jediný účet. Enterprise tarif se dále dělí na dva, které se liší hlavně ve filozofii nasazení. Hosted Storage je klasická cloudová služba s možností až 100 uživatelů s celkovým 1TB. Kdežto tarif Private Cloud, je spíše hybridní řešení, protože data jsou uložena na zákazníkových serverech. Obr. 1 a Obr. 2 názorně ukazují hlavní rozdíly.

SpikeOak podporuje platformy Windows, Mac OS, iOS, Android a Linux, podpora mobilních platforem Windows Phone, nebo BlackBerry není zahrnuta, ale je zde veřejné API, které tuto skutečnost může odstranit.

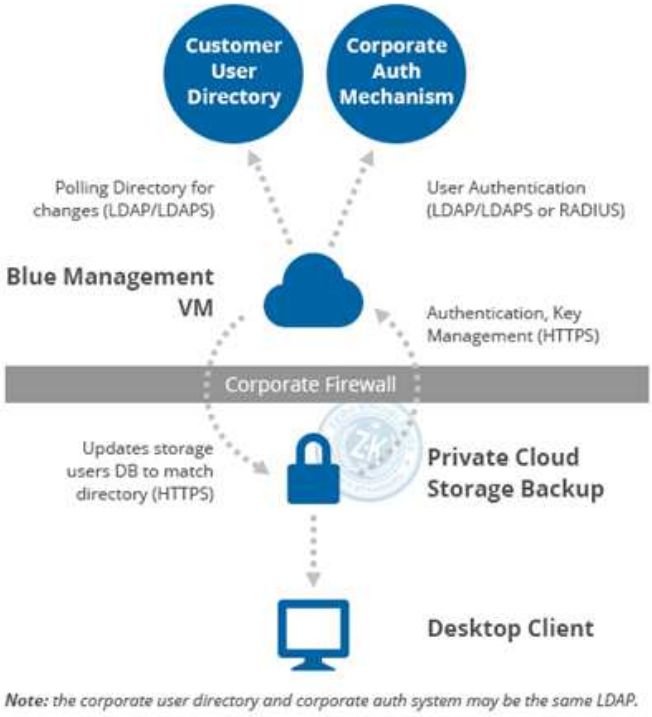
Registrace služby probíhá standardní metodou. Během registrace se přidá první zařízení. Pro účely přihlašování slouží zvolené uživatelské jméno.

Tarif	Cena [$/rok]	Úložný prostor [GB]
Plus Plan	100	100+
Hosted Storage	600	100
Cloud	249.99	250
Cloud	550	1000
Cloud	784-7962	1000+

Tabulka 13. Přehled testováni rychlosti služby SpikeOak

3.5.1. Bezpečnost

Další služba, která si zakládá na vyšší míře zabezpečení. Pro ochranu soukromých dat se řídí svým vytvořeným sloganem „Zero Knowledge“, tudíž služba se snaží neznat obsah souborů.



Obrázek 1. Tarif Hosted Storage [12]

Samotná komunikace zpráv probíhá přes zabezpečený kanál pomocí SSL. Klient šifruje svá data na svém počítači pomocí AES-256 kódování. Pro přenos klíčů je zde použita RSA-3072 šifra. Rozdíl v zabezpečení proti konkurenční Wuale je v tom že zde nalezneme webové rozhranní.

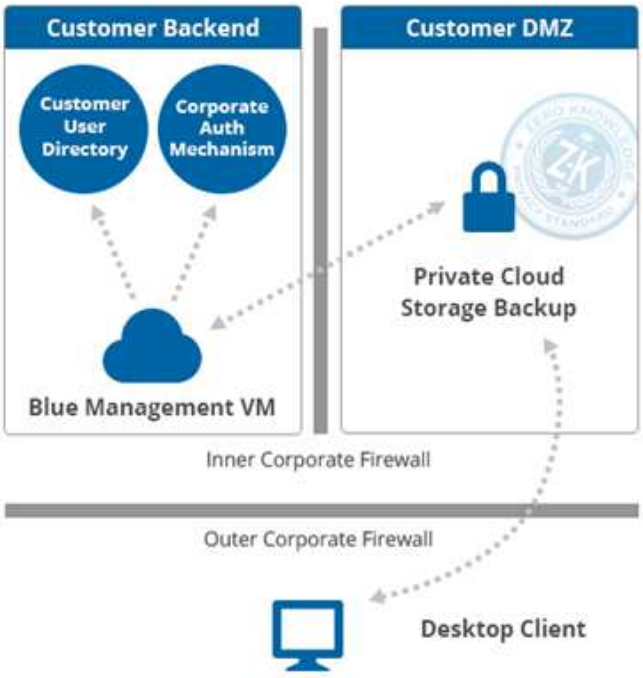
Je velice zarážející že u služby se nachází funkce „Optání na heslo při startu“ které je ve výchozím režimu vypnuta.

3.5.2. Synchronizase a zálohování

SpikeOak není čistě cloudové úložiště, ale spíše zálohovací systém. Tudíž aplikace kromě synchronizace a nahrávání souborů na cloudové úložiště také nabízí zálohování dat, na které je zaměřeno. Tento fakt lze i vypozorovat v podobě nemožnosti synchronizovat libovolnou složku v počítači, ale pouze ty, které si zvolím zároveň jako zálohované. Nicméně zálohovanou složkou, nebo souborem může být cokoliv na kterémkoliv disku. Další kompenzací zmíněného problému s přímou synchronizací je umístění složky „SpikeOak Hive“, která slouží jako synchronizační složka pro libovolné soubory do ní přetažené. Jedná se o stejnou funkcí jako u konkurenčního SugarSync a jejich „Magic Briefcase“.

3.5.3. Sdílení

Možnosti sdílení jsou zde odlišné nežli u konkurence. Klasické veřejné sdílení zde není obsaženo. Je zde možnost vytvořit veřejný odkaz na libovolný soubor v zařízení. Tento odkaz je poté platný následující tři dny a poté se deaktivuje. Pokud chce uživatel sdílet



Obrázek 2. Tarif Private Cloud [12]

nějaký soubor, nelze ho standartně označit a sdílet. Uživatel musí vytvořit „Shareroom“ s heslem a případně odeslat pozvánky jiným uživatelům SpiderOak. Výhodou, ale je, že pro vstup do „Shareroomu“ se cizí uživatel nemusí registrovat, stačí když mu tvůrce Shareroomu odešlo jméno a heslo patřící k příslušnému Shareroomu.

Nevýhodou služby je malá jazyková podpora.

3.5.4. Webový klient

Webové rozhraní služby je v celku přehledné a nabízí základní funkce.

Základními možnostmi jsou prohlížení obsahu na úložišti s možností stáhnutí jednotlivých souborů, nebo jako celku. Nahrávání souborů nicméně nelze. Další možností je zobrazení a definovaní zařízení, které s cloudovou službou pracují. Je zde možné deaktivovat některé s používaných zařízení, ale přidat nové zařízení nelze. Další záložkou je zobrazení sdílených věcí a poslední je nastavení vlastního účtu.

3.5.5. Lokální klient

Klient má aplikaci psanou pomocí Qt knihovny. Na základní stránce nadstandardně zobrazuje veškerá zařízení, která přistupují k učtu spolu s zaplněním. Zajímavou informací je ukazatel stavu propustnosti sítě.

Poměrně nepříjemnou nevýhodou během nahrávání souboru je fakt nemožnost zrušení této operace. Je zde sice možnost pozastavení, ale v případě že uživatel se „překliknul“ musí počkat, až se jeho soubor nahraje a až poté ho odstranit.

Lokální klient i tak je nejsilnější stránkou celé služby a jeví se jako velice komplexní nástroj. Nicméně není tak přehledný jako například u konkurenční Wualy. Klient nabízí široké množství nastavení. Lze zde nastavit rychlost nahrávání souborů. Velice zajímavou funkcionalitou je možnost plánovaní nejen zálohování, ale i synchronizace a sdílení v určitý čas. U zálohování pak lze nastavit maximální velikost nahrávaného souboru, nahrávání podle staří souboru, nebo vyloučení souborů či složek s určitým jménem.

3.5.6. Testování

Služba SpikeOak působila svými časy, že je vytvořena na synchronizaci malých souborů, kde byla nejlepší z měřených služeb. Naopak v otázce velkých a středních souborů patřila k nejhorším. Její průměrná rychlost nahrávání a stahování byla do 1.5Mbit/s resp. 2.5Mbit/s. Tato rychlost je už omezující i pro normálního uživatele. Přehled rychlostí je uveden v Tab. 14 a Tab. 15.

V otázce zatížení byla opět rozporuplná. U středních a velkých souborů se zatížením 5% až 10% není nikterak náročnou. U malých souborů, ale zatížení bylo 60% a 70%, což bylo nejvíce zapříčiněno vytvořením až 4999 tcp spojů. Souhrnné údaje jsou zobrazeny v Tab. 16.

Počet souborů	Celková velikost souborů[MB]	Průměrná rychlost nahrávání [Mbit/s]	Celková doba nahrávání	Průměrná rychlost stahování[Mbit/s]	Celková doba stahování
1	1495	1.3	2h32m	2.4	2h10m
100	331	1.38	32m	2.21	31m
15761	87.8	1.2	10m	0.4	16m

Tabulka 14. Přehled testování rychlostí služby SpikeOak na osobním notebooku

Počet souborů	Celková velikost souborů[MB]	Průměrná rychlost nahrávání [Mbit/s]	Celková doba nahrávání	Průměrná rychlost stahování[Mbit/s]	Celková doba stahování
1	1495	1.483	2h40m	2.45	2h18m
152	501	1.358	1h4m	2.28	33m44s
15761	87.8	0.5	12m	0.4	15m

Tabulka 15. Přehled testování rychlostí služby SpikeOak na referenčním serveru

Počet souborů	Zatížení při nahrávání	Zatížení při stahovánií
1	5%	10%
152	10%	10%
15761	60%	70%

Tabulka 16. Přehled zatížení systému u služby SpikeOak na referenčním serveru

	Google Disk	SkyDrive	DropBox	Ubuntu One
Obecné				
Jazyková lokalizace	●	●	○	●
Integrace do kontextového menu	●	●	○	●
Vlastní nasazení	○	○	○	○
Webová aplikace	●	●	●	●
Klientská aplikace Windows / Linux / Mac OS / Ostatní	●/○/●/○	●/○ /●/○	●/●/●/○	●/●/●/○
Mobilní aplikace Android / Windows Phone / iOS / Ostatní	●/○/●/○	●/●/●/○	●/●/●/●	●/○/●/○
Cloudové úložiště	●	●	●	●
Bezplatná kapacita [GB]	15	7	2	5
Maximální kapacita [GB]	16384	207	100	○
Maximální velikost souboru [MB]	10240	2048	300*4	○
Možnost rozšíření zdarma	○	○	○	○
Synchronizace				
Synchronizace libovolné složky	○	○	○	●
Synchronizace podsložek	●	●	●	○
Synchronizace jednotlivých souborů	○	○	○	○
Pozastavení synchronizace	●	○	●	●
Sledování přenosu dat	●*5	○	○	○
Sdílení				
Veřejné	●	●	○	●
Odkazem	●	●	●	●
Pomocí pozvánky	●	●	●	●
Pomocí mailu	●	●	●	●
Skupinové	●	●	○	○
Nastavení oprávnění	●	●	○	●
Zabezpečení				
Šifrování na straně serveru	○	○	AES	○
Šifrovaní na straně klienta	○	○	○	○
Šifrování přenosu	SSL	SSL	SSL	SSL
Přihlášení pomocí dvoufázového ověření	●	○	●	○
Další funkce				
Obnova smazaných souborů	●	●	●	60 dnů*11
Verzování	●	●*6	●	○
Uložení verzí	30 dnů*7	25 verzí	30 dnů	○
Nastavení rychlosti uploadu	○	○	●	●
Nastavení priorit pro upload	○	○	○	○
Nastavení rychlosti downloadu	○	○	●	●
Rychlosti				
Rychlost uploadu malých souborů [Mb/s]	0,05	0,23	0,25	0.058
Doba uploadu malých souborů	3h 57m	4h12m	15m	2h16m
Rychlost uploadu středně velkých souborů [Mb/s]	4,77	2,83	5,56	9,65
Doba uploadu středně velkých souborů	14m	21m31s	5m	2m30s
Rychlost uploadu velkého souboru [Mb/s]	21,2	4	0,127	0,32
Doba uploadu velkého souboru	10m	58m	90s	3m
Rychlost downloadu malých souborů [Mb/s]	0,08	0,621	-	-
Doba downloadu malých souborů	2h 18m	1h15m	-	-
Rychlost downloadu středně velkých souborů [Mb/s]	26,19	5,315	26,31	-
Doba downloadu středně velkých souborů	3m	51m	2m30s	-
Rychlost downloadu velkého souboru [Mb/s]	58,92	2,8	53,77	11,8
Doba downloadu velkého souboru	4m	1h15m	3m	18m
	Google Disk	SkyDrive	DropBox	Ubuntu One

1 - FreeBSD	5 - Pouze počet přenesených složek a souborů
2 - BlackBerry	6 - Pouze Office soubory
3 - Společný úložný prostor pro služby Google Disk, Gmail a Fotky Google+	7 - Verze souborů z aplikací Google Dokumenty, Tabulky a Prezentace jsou ukládány navždy
4 - Pouze přes webové rozhraní	8 - Ve výchozím nastavení, lze změnit

Obrázek 3. Přehledová tabulka všech služeb

SugarSync	MEGA	TeamDrive	Wuala	cloudMe	SpiderOak	BitTorrentSync	ownCloud
○	●	○	●	○	○	○	●
●	○	○	○	●	●	○	○
○	○	●	○	○	○	●	●
●	●	○	○	●	●	○	●
●/○/●/○	●/(○/○/○)*10	●/●/●/○	●/●/●/○	●/●/●/○	●/●/●/○	●/●/●/●*1	●/●/●/○
●/●/●/●*2	●/○*11/●/●*2	●/○/●/○	●/○/●/○	●/○/●/○	●/○/●/○	●/○/●/○	●/○/●/○
●	●	●	●	●	●	○	○
5	50	2	5	3	2	○	○
1000	4096	neomezeno	2048	500	100	○	○
-	neomezeno	-	40960	150	-	○	○*12
●	○	●	●	○	●	○	○
●	●	●	●	●	●	●	●
●	●	○	●	○	●	○	○
●	●	○	●	○	●	○	○
○	●	●	●	●	●	●	○
●	●	●	●	●	●	●	●
●	●	○	●	●	●	○	○
●	●	●	●	●	●	○	●
●	○	●	○	●	○	○	○
●	○	●	○	●	○	○	●
●	●	●	●	●	●	●	○
○	●	●	●	○	●	●	○
AES-128	○	○	AES-128	○	AES-256	○	●
○	RSA-2048, AES-128	RSA-2048, AES-256	RSA-2048, AES-128	○	RSA-3076, AES-256	AES-128	○
SSL (3.3) + TLS	SSL	○	SSL	SSL	SSL	○	○
○	○	○	○	○	○	○	○
	●	●		●		30 dnů*13	●
●	○	●	●	○	●	●	●
30 dnů	○	neomezeno	10 verzí	-	-	30 dnů*8	-
●*9	●	○	●	●	●	●	●
●	○	○	●	○	○	○	●
○	○	○	●	●	○	●	●
0,04	-	1,17	0.7	0,1	0,5	2,34	0,012
1h50m	-	10m	30m	3h55m	12m	5m	1d7h20m
7,15	1,32	8,6	2,2	4,46	1,358	22,9	4,93
7m15s	51m	8m	13m13s	17m	1h4m	3m	13m
-	63,28	9,09	28,5	-	1,483	33,5	28,92
-	3m	22m	6m	-	2h40m	6m	7m
0,33	-	1,31	2	-	0,4	2,3	2,68
5h13m	-	9m	7m	-	15m	5m	1m30s
2,67	6,34	20,35	10,8	19,89	2,28	22,64	26,94
30m41s	11m	4m	6m58s	2m30s	33m44s	3m	3m
2,9	35,6	10,98	7,3	-	2,45	28,61	43
1h20m	6m	18m	30m41s	-	2h18m	7m	5m
SugarSync	MEGA	TeamDrive	Wuala	cloudMe	SpiderOak	BitTorrentSync	ownCloud

9 - Pouze relativní nastavení
10 - Zatím ve vývoji (17.11.2013)
11 - bez 100% garance
12 - přes webové rozhraní 8MB jinak podle konfigurace
13 - Dobu lze nastavit

4. Koncepce distribuovaného úložiště

Tato část textu pojednává o základní představě navrhovaného systému distribuovaného uložiště.

Systém je tvořen ze tří základních prvků
- Klientská aplikace
- Access server
- Datové centrum

4.1. Klientská aplikace

Klientská aplikace je tvořena třemi částmi. První část se stará o komunikaci s hostitelským souborovým systémem. Druhá část obstarává komunikaci s access serverem. Poslední třetí část je zodpovědná za práci s metadaty.

Práce s metadaty je realizována pomocí samostatné knihovny. Tato knihovna udržuje informace o adresářové struktuře a vlastních souborech v ní obsažených.

4.2. Access server

Access server (AS) odstínuje klientskou aplikaci od datového centra a zprostředkovává komunikaci mezi nimi. AS tedy tvoří přístupový bod pro klientské aplikace, které se do systému připojují. Zároveň poskytuje funkce potřebné pro sdílení a historii uživatelských akcí.

4.3. Datové centrum

Datový subsystém tvoří vlastní úložiště dat. Skládá se z jednotlivých datových center, která jsou spojena do strukturované sítě pomocí DHT. Každé jednotlivé centrum je schopno obsluhovat požadavky AS.

4.4. Komunikace

Komunikace mezi jednotlivými prvky systému je tvořena pomocí socketů. Pro výměnu informací mezi jednotlivými prvky systému se využívá informačních zpráv, které jsou specifické pro jednotlivý subsystém a prováděnou operaci.

V rámci celého systému neexistuje žádná databáze uživatelských identifikátorů.

Pro identifikaci uživatelů se používá jednoznačný identifikátor, který je generován klientskou aplikací při prvním přihlášení do systému. Tento identifikátor je vytvořen pomocí hashovaní funkce SHA-2 s celkovou délkou 80-bytů. Pro určení lokalizace souborů

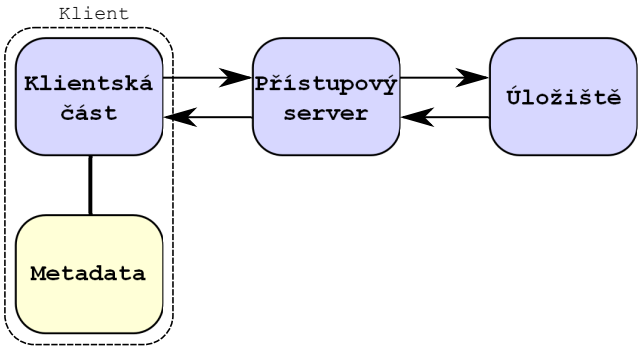
v rámci DHT datového centra se používá prvních 40-bytů identifikátoru daného souboru.

4.5. Zabezpečení systému

Jednotlivé soubory jsou rozděleny do chunků o maximální velikosti 4MB, které jsou následně zabezpečeny s využitím šifrovacího algoritmu AES-128. U každého souboru lze nastavit počet replikovaných souborů v rámci datového centra.

Komunikace je mezi jednotlivými prvky systému zabezpečena na úrovni protokolu TLS s PGP mechanizmem.

Výsledná celková koncepce systému je znázorněna na Obr. 4.



Obrázek 4. Koncepce distribuovaného úložiště

5. Analýza a návrh řešení

V následujícím textu provedeme návrh subsystému, který se stará a samotné uložení dat. Jednotlivé uzly v rámci subsystému budeme nazývat datová centra. Dále se podíváme na základní možné výpadky v rámci systému.

5.1. Propojení uzlů

První otázkou při návrhu konceptu bylo, jakou koncepci zvolit pro propojeni jednotlivých datových center. Nabízí se dvě základní možnosti. Buď vytvořit strukturovanou síť a nebo nestrukturovanou síť. Nicméně v našem případě nestrukturovanou koncepci sítě nemůžeme použít. Důvodem je základní nedostatek algoritmů co tyto sítě implementují. Důvodem je především nezajištění, že existující data budou skutečně nalezena. Kdežto strukturované sítě naleznou existující data v konečném počtu kroků. Pro strukturovanou síť jsou ideální distribuované hašovací tabulky (DHT). Mají výhodu v efektivním vyhledávání v síti. Nevýhodou je naopak připojování respektive odpojování uzlů, což souvisí s změnami směrovacích tabulek. Bylo tedy nutné najít vhodné řešení pro implementaci DHT.

5.1.1. Vlastní DHT knihovna

Nejlepším možným řešením by bylo navrhnout si svojí vlastní DHT knihovnu. Taková to knihovna by byla přizpůsobena přesně požadavkům distribuovaného úložiště. Takovýto postup je velice časově a implementačně náročný a tudíž, pro tuto možnost jsem se nerozhodl.

5.1.2. DHT knihovna

Výběr DHT knihovny nejprve směřoval k výběru takové, která by implementovala Kademlii. Kademlie poskytuje lepší vlastnosti pro systém v porovnání s nejpoužívanějšími řešeními jako je Chord, nebo Pastry. Zmíněnými vlastnostmi je především snadný postup pro připojení respektive odpojení uzlu. Tato operace je v případě Chordu značně obtížnější. Pastry zase naopak poskytuji horší směrovací výkon.

Bohužel jak se ukázalo, v současnosti není mezi dostupnými DHT knihovnami implementovanými v jazyku C velký výběr. Tudíž výběr nakonec byl, zda vůbec nějaká knihovna půjde použít. Jako hlavní kandidáti byly tyto čtyři:

- MaidSafe [13]
- OpenP2P [14]
- Telehash [15]
- Chimera [16]

Ze začátku bylo zamýšleno použít velice robustní a komplexní knihovnu MaidSafe-DHT. Tato knihovna nabízí průchody přes NAT zařízení, šifruje veškerou komunikaci

a je multiplatformní. Bohužel po bližším zkoumáním jsem jí musel opustit. Důvodem je, že knihovna MaidSafe-DHT už není 3 roky vyvíjena a neposkytuje skoro žádnou dokumentaci použití. Další nevýhodou je nutnost použít další knihovny od skupiny Maidsafe. Maidsafe-DHT ve vývoji přešla na Maidsafe Rating, která ovšem je už začleněna do celého projektu Maidsafe, tudíž manipulace s ní je ještě více problematičtější.

Další alternativou bylo požití knihovny OpenP2P. Používá kryptografii, má implementovaný i průchod přes NAT. Nicméně opět chybí dokumentace a její kód se v době psaní této práce měnil takřka každý den.

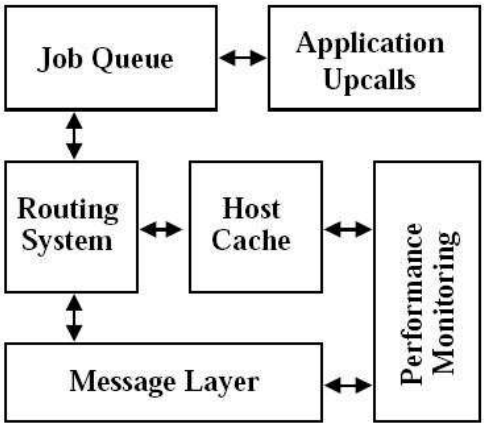
Třetí možností bylo použití knihovny Telehash. Minimalistická knihovna používající kryptografii nad Kademlií. Nicméně implementace v jazyce C nebyla v době výběru plně dokončena.

Nakonec jsem tedy zvolil velice starou DHT knihovnu chiméra, kterou už dříve použil při implementaci vlastního systému kolega Ing. Tuček [17]. Knihovna má alespoň nějakou dokumentaci. Nicméně knihovna toho i moc nenabízí a bylo ji tedy nutno upravit. Knihovna je postavena na architektuře Pastry. Její vývoj pro jazyk C byl ukončen už v roce 2006 a byl přesměrován pouze na jazyk JAVA.

Knihovna Chimera umí v podstatě pouze vlastní DHT. Jak bylo zmíněno je napsána v jazyce C a její implementace je v celku minimalistická. Knihovna je částečně multiplatformní. Běží na operačních systémech Linux a Windows. Nicméně se ukázalo, že i tato knihovna obsahuje chyby, které bylo nutné opravit. Popis oprav je popsán v kapitole realizace. Dále tato knihovna používá jako pro svůj stavový prostor 160-bitový hash vygenerovaný pomocí hašovací funkce SHA-1. Kolize jsou tedy reálné s pravděpodobností ukazující na Obr. 5. Architektura knihovny Chimera je znázorněna na Obr. 6.

Number of **32-bit** hash values	Number of **64-bit** hash values	Number of **160-bit** hash values	**Odds of a hash collision**	
77163	5.06 billion	1.42×10^{24}	1 in 2	
30084	1.97 billion	5.55×10^{23}	1 in 10	
9292	609 million	1.71×10^{23}	1 in 100	Odds of a full house in poker 1 in 693
2932	192 million	5.41×10^{22}	1 in 1000	Odds of four-of-a-kind in poker 1 in 4164
927	60.7 million	1.71×10^{22}	1 in 10000	
294	19.2 million	5.41×10^{21}	1 in 100000	Odds of being struck by lightning 1 in 576000
93	6.07 million	1.71×10^{21}	1 in a million	Odds of winning a 6/49 lottery 1 in 13.9 million
30	1.92 million	5.41×10^{20}	1 in 10 million	
10	607401	1.71×10^{20}	1 in 100 million	Odds of dying in a shark attack 1 in 300 million
	192077	5.41×10^{19}	1 in a billion	
	60740	1.71×10^{19}	1 in 10 billion	
	19208	5.41×10^{18}	1 in 100 billion	
	6074	1.71×10^{18}	1 in a trillion	
	1921	5.41×10^{17}	1 in 10 trillion	
	608	1.71×10^{17}	1 in 100 trillion	Odds of a meteor landing on your house 1 in 182 trillion
	193	5.41×10^{16}	1 in 10^{15}	
	61	1.71×10^{16}	1 in 10^{16}	
	20	5.41×10^{15}	1 in 10^{17}	
	7	1.71×10^{15}	1 in 10^{18}	

Obrázek 5. Pravděpodobnost kolizí hashovací funkce SHA-1. [18]



Obrázek 6. Architektura knihovny Chimera [16].

5.2. Komunikace

Další otázkou návrhu byla komunikace mezi jednotlivými datovými uzly. Knihovna Chimera nám poskytne vybudování DHT sítě a šíření informací k jednotlivým uzlům. Koncepce kdy by jednotlivá datová centra posílaly svoje data přímo přes Chimeru jsem zavrhl. Důvodem je především, že Chimera používá pro přenos maximálně zprávu o velikosti 65kB a využívá UDP sockety u kterých ovšem má implementovaný proces potvrzovaní jednotlivých posílaných zpráv. Pokud by tedy datové centrum chtělo poslat větší soubor než 65kB musela by se implementovat další logiku, které by zajišťovala správnou funkcionalitu. Navíc UDP protokol je méně vhodný pro přenos souborů než TCP.

Mmou zvolené řešení pro přenos datových souborů je následující. Pomocí DHT sítě se nalezne požadovaný server pro příslušnou operaci, který se spojí s server, který požadavek poslal a přenos se provede pomocí TCP spojení.

Dalším aspektem v návrhu komunikace je bezpečnost. Knihovna Chimera má implementovány pouze čisté UDP sockety bez jakéhokoliv zabezpečení, bylo tedy nutné vybrat vhodné řešení pro zabezpečení komunikace. To bylo provedeno především ve spolupráci s Bc. Tomášem Dyntarem. Jako nejlepší návrh je použití šifrování s ověřením pomocí PGP.

Bylo tedy nutné vybrat vhodnou knihovnu pro šifrování komunikace. V úvahu připadala pouze knihovna GnuTLS [19] s širokou podporou. Široce používaná knihovna OpenSSL koncepci PGP zatím nepodporuje.

5.3. Databáze

Díky navržené koncepci distribuovaného úložiště, bylo jasné, že bude nutné implementovat nějakou formu databáze identifikatorů chunků na datovém centru. Pro účely datového centra bylo tedy nutné nalézt vhodné efektivní řešení této databáze. Jako nejlepší možnost se jeví použití minimalistické databáze SQLite [20]. Není nikterak robustní a

náročná jako například databáze PostrgreSQL, nicméně stále si zachovává dostatečnou funkcionalitu a rychlost.

5.4. Funkce systému

Následující text popisuje navrhnuté chování datových center.

5.4.1. Ukládání chunku

Ukládání dat je jednou z primárních funkcí datového centra. Vlastní návrh mechanismus ukládání chunku se provádí takto:

1. Access server zašle na některý z datových center z DHT sítě požadavek na ukládání chunku.
2. Uzel v DHT síti požadavek přijme a přepošle tento požadavek síti DHT na příslušný uzel na který se mají data uložit.
3. Tento uzel požadavek zpracuje a připojí se k Access Serveru, který je identifikován v požadavku a vyžádá si příslušná data identifikována hashem chunku v požadavku.

Tento koncept má i svojí nevýhodu a to v tom, že Access Server musí být viděn pro všechny uzly DHT sítě.

5.4.2. Zaslání chunku

Další z primárních funkcí systému je načítání chunků a poslání příslušnému tazateli. Tento proces se provádí takto:

1. Access server zašle na některý z datových center v DHT síti požadavek na čtení chunku.
2. DHT uzel požadavek přijme a přepošle sítí DHT k datovému centru, který tento chunk vlastní.
3. Tento uzel požadavek zpracuje a připojí se k Access Serveru, který je identifikován v požadavku a zašle příslušný chunk.

5.4.3. Inicializace

Inicializace probíhá při startu systému. Načte se databáze obsažených chunků na datovém centru. Provede se spuštění file serveru, který zajišťuje přenos chunků. Dále se provede inicializace s připojení DHT sítě Chimera. Poté se provede spuštění funkcí na mazání chunků a získávání chunků. Pokud je spuštění provedeno s příznakem připojení, provede se postup začleňování, tento postup je níže popsán.

5.4.4. Připojování uzlů

Přidání datové uzlu se provede automaticky po spuštění nového uzlu se správnými parametry. Připojovaný uzel musí znát alespoň jedno datové centrum ze sítě DHT. Chimera provede začlenění do stavového prostoru. Následně se provede stažení všech identifikátorů, které jeho stavový prostor vlastní. Poté si zažádá o všechny identifikátory replik, které mu jsou nyní přiděleny a ty si pak nechá zaslat.

Sousedé nově připojeného datového centra si zaktualizují záznamy sousedů . Poté co připojované datové centrum si stáhne všechny chunky, vymazají si ty chunky, které byly nově přiděleny připojovanému datovému centru.

5.4.5. Aktualizace časového záznamů chunků

Aktualizace časového záznamu chunků se provádí po výzvě access serveru, který tak učiní pomocí zprávy na některý z datových center DHT sítě. Datové centrum požadavek přijme a zpracuje a vyžádá si seznam identifikátorů k aktualizovaným chunkům. Poté postupně zasílá aktualizační zprávy s identifikátorem chunku k jednotlivým datovým centrům, které obsahují chunky s tímto identifikátorem. Ty si tyto chunky zaktualizují a provedou zaslání potvrzení o úspěšné aktualizaci uzlu, který aktualizační zprávu vyslal.

5.4.6. Mazání chunků

Mazání chunků je dvojího typu:

- Periodické - Mazání chunků se provádí periodicky na každém uzlu DHT sítě po 24hodinách. Při této operaci jsou mazány veškeré chunky z disku i databáze, které mají časový záznam starší než doba, která byla nastavena v našem případě 90 dnů.
- Na vyžádání - Je prováděno při připojování uzlu, nebo při odpojování/pádu datového centra. Jsou mazány ty chunky, které už nepřísluší danému datovému centru.

5.4.7. Replikace chunků

Replikace chunku je prováděna po operaci uložení chunků v případě, že chunk je vyžadován alespoň v jedné replikaci. Server, který chunk uloží odešle požadavky na uložení replik na sousední uzly. Ty tento požadavek zpracují a vyžádají si příslušný chunk, který následně uloží.

5.4.8. Výpadek uzlu

Pokud nějaké datové centrum vypadne. DHT sama díky knihovně Chimera opraví stavový prostor a směrovací tabulky. Jednotlivá datová centra si změní identifikátor u replik chunků z odpojeného na identifikátor datového centra, který ho nahradí. Datové centrum, které bude nahrazovat odpojovaného si jeho chunky přivlastní. Poté je nutné aby datová centra, kterých se to týká požádali o nové repliky, které jsou jim nyní po výpadku přiděleny.

5.4.9. Odpojení

Při procesu odpojení datové centrum přejde do stavu, že už neukládá žádné chunky. Tyto dotazy směruje na uzel, který ho nahradí po odpojení. Ostatní operace přestává vykonávat.

Dále až nebude zpracováván žádný požadavek na ukládání, pošle všechny svoje data, které nemají repliku uzlu kterému po odpojení připadne stavový prostor odpojovaného. Po této operaci se provede odpojení z Chimery a ukončení všech funkcí a samotného programu.

5.4.10. Hospodárnost místa

Je jasné, že zvolený koncept není příliš hospodárný k úložnému prostoru. Každá změna je ukládána ve formě nového souboru. Nicméně je žádoucí, aby na jednom datovém centru nebyly uloženy stejné soubory víckrát.

V případě kdy uživatel prostřednictvím access server se pokusí uložit data s identifikátorem, který je už v databázi uložen bude informován, že tyto data jsou už uložena.

V případě, že dva uživatelé budou chtít uložit ty samá data, která budou zašifrovaná nelze ošetřit, aby tyto data byla pouze jednou na ukládaném datovém centru. Důvodem je, že data budou mít jiný identifikátor, tudíž jeví se jako dva rozdílné soubory.

V případě stejných nezašifrovaných dat, nebo stejných dat zašifrovaných stejným klíčem bude situace jiná. Datové centrum takovýto chunk uloží. Poté pokud nalezne v databázi stejný 256-bitový prefix identifikátoru bude provedena mezi nově uloženým a nalezeným identifikátorem srovnání zda jsou stejné. Pokud ano, nově vytvořený chunk se smaže a vytvoří se odkaz na nalezený.

5.4.11. Spravování místa

Na každém uzlu bude možné nastavit minimální možnou kapacitu volného místa. Pokud tato kvóta bude překročena, na datový uzel tedy nebude možné dále ukládat data.

Díky rovnoměrnému rozložení generování hashe pomocí SHA-2 budou data rovnoměrně ukládána v rámci DHT sítě. Tudíž zaplnění jednotlivých datových center bude takřka stejné. Díky těmto aspektům je jasné, že nejmenší disk v síti bude zaplněn nejrychleji. Je tedy nutné navrhnout algoritmus, který by v případě zaplnění přesměroval požadavky na ukládání na některý jiný uzel, který požadovanou úložnou kapacitu vlastní. Tento koncept je možné řešit takto:

- Pomocí odkazů - jedná se o proces, kdy uzel, který nemá dostatek místa se dotáže svých sousedů, zda není místo u nich. Pokud ano, předá požadavek na uložení těmto sousedům a u sebe v databázi si vytvoří odkaz na souseda u kterého bude chunk uložen.
- Přeskupeni stavového prostoru - Samotná datová centra si v závislosti na dostupné kapacitě budou přerozdělovat velikost stavového prostoru. Vyřešení toho konceptu je nad rámec diplomové práce.

5.4.12. Monitoring

Server trvale sleduje kolik aktivních připojených klientů a kolik aktivních připojení má jeho server. Dále je při každém požadavku na uložení kontrolována dostupnost úložného prostoru.

5.4.13. Přepsání chunku

Proces kdy je vyžadováno přepsání chunku se provádí takto:

1. Access server zašle na některý z datových center z DHT sítě požadavek na přepsání chunku.

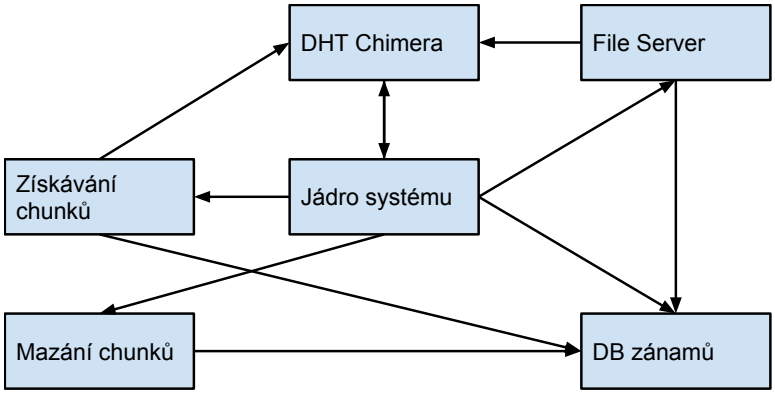
2. Datové centrum v DHT síti požadavek přijme a přepošle tento požadavek síti DHT na příslušný uzel na kterém se mají data přepsat.
3. Toto datové centrum požadavek zpracuje a připojí se k Access Serveru, který je identifikován v požadavku a vyžádá si příslušná data identifikována hashem chunku v požadavku a následně chunk přepíše.

5.4.14. Přepis replik chunků

Přepsání replikace chunku je prováděna po operaci přepsání chunků v případě, že chunk vlastní alespoň jednu replikaci. Server, který chunk přepíše odešle požadavky na přepis replik na příslušné sousední datová centra. Ty tento požadavek zpracují a vyžádají si příslušný chunk, který následně přepíší.

5.5. Celkový pohled na systém

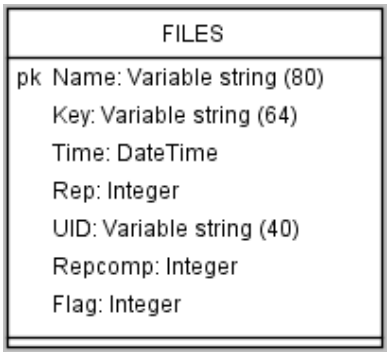
Z předešlého popisu funkcí se nyní podíváme na celkový návrh jednotlivých modulů. Jednotlivé moduly a jejich propojení charakterizuje Obr. 7.



Obrázek 7. Tabulka záznamů chunků

5.5.1. DB server

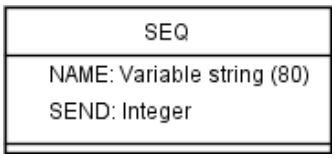
Databáze zpravuje záznamy všech uložených chunků v formátu zobrazeném na Obr. 8.



Obrázek 8. Základní architktura navrhovaného systému

- Name: je identifikátor chunku
- Identifikátor chunku bez uživatelských informací
- Time: poslední čas přístupu k chunku
- Rep: je číslo kolikrát má být záznam replikován
- UID: je identifikátor vlastníka chunku
- Repcomp: počet úspěšně uložených replikací
- FLAG: zda je soubor fyzicky uložen

Dále slouží jako dočasné úložiště identifikátorů chunků u kterých má být zaktualizován časový záznam. Formát tabulky je zobrazen na Obr. 9.



Obrázek 9. Tabulka pro aktualizaci záznamů chunků

- NAME: je identifikátor chunku
- SEND: počet pokusů o aktualizaci časového záznamu

5.5.2. Chimera

Stará se o vlastní DHT síť. Tedy o propojení s ostatními datovými centry pomocí kterých se vyhledávají datová centra respektive chunky a rozesílají informačních zprávy. Formát těchto navrhovaných informační zpráv je zobrazen v Tab. 17.

Typ operace	Formát
Uložení dat	Identifikační kód|IP adresa tazatele|Port tazatele|Velikost dat|Počet replik|Identifikační klíč
Přepsání dat	Identifikační kód|IP adresa tazatele|Port tazatele|Velikost dat|Počet replik|Identifikační klíč
Načtení dat	Identifikační kód|IP adresa tazatele|Port tazatele|Identifikační klíč
Aktualizace záznamů	Identifikační kód|IP adresa tazatele|Port tazatele|Identifikační klíč
Uložení repliky	Identifikační kód|IP adresa tazatele|Port tazatele|Velikost dat|Počet replik|Identifikační klíč|Klíč vlastníka
Žádost o identifikátory replik	Identifikační kód|IP adresa tazatele|Port tazatele|Počet úrovní o které je žádáno|Úrovně replik
Žádání o repliku	Identifikační kód|IP adresa tazatele|Port tazatele|Identifikační klíč
Přepsání repliky	Identifikační kód|IP adresa tazatele|Port tazatele|Velikost dat|Počet replik|Identifikační klíč|Klíč vlastníka
Oznámení o uspěšné aktualizaci	Identifikační kód|Identifikátor aktualizační tabulky|Identifikační klíč
Žádost o svoje identifkátory	Identifikační kód|IP adresa tazatele|Port tazatele|Klíč tazatele

Tabulka 17. Formát dotazovacích zpráv

5.5.3. File Server

File server je spuštěn ihned po startu systému a obsluhuje všechny klienty kteří se k němu připojí. Klienti jsou dvojí, buď Access Server nebo jiné datové centrum. Access Server vznáší na File Server požadavky na čtení, zápis nebo aktualizaci časových záznamu chunků. File Server tyto požadavky zpracuje a pošle na příslušný uzel pomocí DHT síte požadavek Access Serveru. Pokud se jedná o komunikaci s datovým centrem. File server obstarává přímou výměnu dat mezi tímto datovým centrem.

5.5.4. Jádro systému

Je zodpovědné za spuštění celého systému. Dále obstarává aktualizaci tabulek sousedu spolu s obsluhou příkazů z konzole.

5.5.5. Mazaní chunků

Tento modul je zodpovědný za promazávání chunků z datového centra pokud je záznam starší než zvolená kvóta v našem případě 90 dnů. Mazání je prováděno vždy jednou denně.

5.5.6. Získávání chunků

Modul který se stará o rozesíláni informačních zpráv, které obsahuji informace, které repliky mají být tomuto uzlu poslány. Jsou vyžadovány všechny chunky, který mají v databázi záznamu příznak FLAG nastaven na 0.

5.6. Komunikační model

Datové centrum je konceptuálně místo pro uložení dat. Komunikace tedy probíhá s access serverem a mezi jednotlivými datovými centry DHT sítě. Veškerá tato komunikace bude realizována pomocí TLS spojení s PGP ověřením. Spojení mezi access serverem a datovým centrem bude probíhat na TCP spojeních. Datové transfery mezi uzly datové centra budou probíhat na TCP spojeních. Informativní zprávy v rámci datového centra budou šířeny pomocí DHT sítě využívající UDP spojení.

5.6.1. Struktura komunikace s access serverem při uložení chunku

Příchozí zpráva:
- Typ požadavku
- IP adresa Access Serveru
- Port Access serveru
- Velikost ukládaného chunku
- Počet replikací
- Identifikátor chunku

Odpověď:
- Typ požadavku
- Stavový kód
- Identifikátor

Příchozí zpráva:
- Potvrzení operace
- Poslání chunku

Odpověď:
- Potvrzení přijetí

5.6.2. Struktura komunikace s access serverem při zaslání chunků

Příchozí zpráva:
- Typ požadavku
- IP adresa Access Serveru
- Port Access serveru
- Identifikátor chunku

Odpověď:
- Typ požadavku
- Identifikátor
- Velikost čteného chunku
- Stavový kód

Příchozí zpráva:
- Potvrzení operace

Odpověď:
- Poslání chunku

Příchozí zpráva:
- Potvrzení operace

5.6.3. Struktura komunikace s access serverem při update chunků

Příchozí zpráva:
- Typ požadavku
- IP adresa Access Serveru
- Port Access serveru
- Identifikátor souboru
- Velikost souboru

Odpověď:
- Potvrzení operace

Příchozí zpráva:
- Poslání souboru

Odpověď:
- Potvrzení přijetí

5.7. Výpadky

Výpadky jsou nedílnou součástí každé aplikace, vzlášť pokud se jedna o distribuovanou aplikace. Následujicí text se zaměřuje na základní výpadky v rámci celého navrhovaného systému distribuovaného uložiště. Tyto výpadky mohou být různého rázu jako špatné formáty předávání zpráv, nebo selhání během procesu ověřování pomocí PGP.

5.7.1. Výpadky na straně klienta

Základní výpadky může být dvojího typu:
- Výpadek AS
- Výpadek samotného klienta

Pokud během spojení vypadne klientovi AS, musi mít k dispozici udaje k dalšímu AS. Pokud klient nevytvoří spojení s žádným AS o kterých má záznam klient začne pracovat v režimu offline. Bude nicméně v určitých intervalech testovat, zda je už nějaký z AS dostupný. Poté klient přejde opět do režimu online a provede zápis všech změn na distribuované uložiště.

Pokud spadne celá aplikace, nebo systém aplikace musí být opět manuálně spuštěna.

5.7.2. Výpadky na straně access serveru

Základní výpadky jsou trojího typu:
- Výpadek AS
- Výpadek klienta
- Výpadek datového centra

V případě výpadku AS si ostatní AS aktualizuji záznam v směrovací tabulce.

V případě, kdy nastane výpadek klienta během procesu je následně celý proces anulován. Následně AS, který byl v tuto dobu s klientem ve spojení rozešle informaci o odpojení klienta ostatním AS.

V případě pádu datového centra má AS opět směrovací tabulku s dalšími datovými centry. Pokud je výpadek během procesu je proces anulován a začíná se nově s jiným datovým centrem.

5.7.3. Výpadky na straně datového centra

Výpadky na straně datového centra jsou řešeny pomocí zmíněných funkcí, které byly popsány v předešle sekci. Implementace těchto funkcí je dále popsána v kapitole Realizace 6.

6. Realizace

V analýze jsme se věnovali návrhu. Nyní si popíšeme implementaci pilotní verze navrhovaného subsystému. V pilotní verzi nejsou implementovány veškeré návrhy z předchozí analýzy. Nicméně pilotní verze je schopna provádět základní operace.

V této implementaci tedy nalezmene následující funkce:

- Čtení chunků
- Zapis chunků
- Replikace chunků
- Aktualizace časových záznamů chunků
- Připojování datového centra
- Hospodárnost místa
- Výpadek datového centra
- Monitoring
- Mazání chunků periodické
- Přepsání chunků
- Přepsání replik chunků

Naopak pro nedostatek času a náročnější implementaci nejsou zahrnuty tyto funkce:

- Zabezpečení DHT chimery pomocí DTLS s PGP
- Spravování místa pomocí odkazů
- Mazání chunků na vyžádání při operaci výpadku, odpojení a připojení datového centra
- Odpojení uzlu

6.1. Vývojové prostředí a implementační jazyk

Jako vývojové prostředí jsem použil platformu Eclipse, konkrétně verzi 4.3.2(Kepler)[gd21]. Jedná se z mého hlediska o jednoduché a přehledné prostředí, které svými funkcemi plně pokrylo požadavky na psaní kódu. Jako implementační jazyk jsem použil jazyk C, konkrétně práce se snažila dodržet standart C90. Tento jazyk jsem použil hlavně z důvodů, že všechny hlavní potřebné knihovny byly napsány v jazyku C a větší zkušeností s programováním. Nevýhodou nicméně je ochuzení o vlastnosti OOP.

Implementace byla napsána na GNU/Linux, konkrétně na Ubuntu 12.04LTS 32bit[gd22] z důvodu širší podpory pro tento druh programů a vyšší pohodlnosti, která byla zapříčiněná hlavně většími zkušenostmi psaní v tomto systému. Nicméně je důležité zmínit, že systém byl psán s ohledem na přenositelnost na jiné systémy (Windows).

6.2. Komunikace

Jak již bylo v návrhu řečeno, bylo nutné komunikaci zabezpečit. Samotné propojeni AS a File Serveru datových center je realizováno na zašifrovaném TCP spoji. Šifrování je

prováděno pomocí GnuTLS knihovny. Komunikace v DHT síti realizovaná na protokolu UDP nebyla zabezpečena z těchto důvodů:

- Nutné přepracování komunikačního schématu.
- Změna předávání velikosti zpráv s případným rozdělováním zpráv (DTLS pomocí GnuTLS nesmí být fragmentováno, tudíž velikost zpráv je limitována na obvyklou hodnotu 1500 bajtů. Nicméně chimera používá zprávy o maximální velikosti až 65kB).
- Celková časová náročnost na úpravu.

Je důležité zmínit, že musel být opraven celý proces ověřování životnosti uzlu v knihovně Chimera. Tento proces pouze zasílal UDP packety na stranu ověřovaného nicméně nereagoval pokud žádná odpověď nepřišla. Bylo tedy nutné vytvořit jednoduchou a časově nenáročnou opravu.

Byla tedy vytvořena ověřovací služba, která na všechny uzly z tabulky sousedů i směrovací tabulky posílal ověřovací zprávy. Pokud ověřovaný uzel neodpoví na tři volání s maximální časem odpovědi 2 sekundy je prohlášen za neaktivní uzel a je z tabulky sousedů, nebo směrovací tabulky odstraněn.
Některé další problémy s Chimerou jsou, nekompatibilita mezi 32bit a 64bit systémy. Dále široce spoléhá na funkčnost DNS zaznámů, protože pro identifikaci uzlů používá doménová jména.

6.3. Implementace DHT

Chimera nám zajistí vytvoření strukturované DHT sítě. Zaručí nám tedy, že zprávy budou odesílány k nejbližšímu uzlu s identifikátorem zprávy.

Napojení celého systému Chimera je následující. Nejprve se spustí inicializaci Chimery, poté následuje vytvoření identifikačního čísla v rámci Chimery, kde se jedná o 160bit hash který je vytvořeny pomocí SHA-1. Poté následuje registrace callback funkcí v našem případě jsou to tyto tři:

- `chimera_forward (state, forwarding);`
- `chimera_deliver (state, delliver);`
- `chimera_update (state, update_message);`

poté jsou zaregistrovány veškeré použité komunikační zprávy. Jejich názvy jsou následující:

- `FIND_KEY_FOR_SAFE`
  - identifikátor pro uložení chunku
- `FIND_KEY_FOR_RSAFE`
  - identifikátor pro přepsání chunku
- `FIND_KEY_FOR_SEND`
  - identifikátor pro čtění chunku
- `UPDATE CHUNKS`
  - identifikátor pro aktualizaci časového záznamu
- `REP_SAFE`
  - identifikator pro uložení repliky chunku
- `RREP_SAFE`
  - identifikator pro přepsání repliky chunku

- `REP_GSAFE`
  - identifikátor o zaslání repliky
- `GET_MY_RECORD`
  - identifikátor pro záslání svých identifikačních záznamů chunků
- `UPDATE REP`
  - identifikátor aktualizace časových zaznámů replik
- `UPDATED_CHUNKS`
  - identifikátor o potvrzení aktualizace časového záznamu chunku
- `UP_REP`
  - identifikátor o zaslání identifikátorů replik
- `CAN_UPDATE`
  - identifikátor o možnosti provedení zaslání svých identifikačních záznamů

Nakonec se provede připojení ve formě zaslání připojovací zprávy k uzlu ke kterému se připojujeme.

Směrování probíhá následovně. Identifikátor zprávy se nejprve porovná s identifikátory v tabulce sousedů. Pokud spadá k některému z nich je požadavek zprávy zasláno k tomuto uzlu. Pokud není je prohledána směrovací tabulka a zpráva je zaslánu tomu uzlu, který má s identifikátorem nejdelší společný prefix.

Odpojení systému není jako takové v knihovně Chimera implementováno.

Jelikož v navrhovaném systému používáme identifikátory o délce 80-bytů bylo nutné pro směrování v rámci Chimery je oříznout na 40-bytů. Změna hashovací funkce u této knihovny, by byla velice problematická. Tím ovšem nám vznikla pravděpodobnost, že požadavek nebude doručen na správný uzel.

V knihovně Chimera se dále neuvažuje o kolizích identifikátorů uzlů. Dá se, ale konstatovat, že tato situace při nasazení reálného počtu uzlů takřka nenastane. Pravděpodobnost kolizí 160-bit SHA-1 nám už v analýze ukázal Obr. 5.

6.4. Vlákna programu

Navržený koncept systému z analýzy Obr. 8, byl použit i při návrhu. Jednotlivé moduly jsou většinou realizovány pomocí samostatných vláken. Samotná vlákna jsou realizovaná pomocí POSIX vláken, jsou tedy kompatibilní s Unix-like systémy. Na systémech Windows je nativní implementace zajištěna buď pomocí podsystému SFU/SUA[gd23], nebo pomocí balíčků třetích stran například pthreads-s32[gd24].
Trvalá vlákna programu jsou:

- Jádro systému
- Získávání chunků
- Mazání chunku (periodické)
- File Server
- Vlákna chimery

6.4.1. Jádro systému

Realizuje inicializaci a dále zobrazování dostupných údajů. Jako počet připojených klientů, počet aktivních připojení File Serveru, celkovou velikost místa na disku a celkovou volnou kapacitu na disku. Dále umožňuje příkaz na násilné ukončení datové centra.

6.4.2. FileServer

Modul, který běží ve vlastním vlákně po celou dobu běhu systému. Je zde spuštěn server na vlastním portu, který je zadán v konfiguračním souboru. Server se stará o příchozí požadavky AS, nebo jiných datových center. Po úspěšném navázání s klientem je každý požadavek dále předán nově vytvořenému vláknu, který jej zpracuje. Tyto dílčí vlákna využívají metodu s názvem:

```
void *server_handler(void *socket_desc)
```

6.4.3. Mazání chunků

Představuje další jedno samostatné vlákno, které běží po celou dobu běhu systému. Jeho funkce je popsána v sekci 6.10. a využívá metodu s názvem:

```
void * delete_my_old_files (void *parameters)
```

6.4.4. Získávání chunků

Jedná se o module, který během své činnosti prochází v SQLite databázi tabulku FILES se záznamy chunků. Hledá takové identifikátory, které mají příznak FLAG nastaven na 0. O tyto záznamy si poté zažádá prostřednictvím Chimera zprávy kde jako cíl určení je UID a jméno Chunku je NAME.

```
chimera_send (state, key_sq, REP_SAFE2, strlen(sql)+1, sql);
```

Vždy je vykonáváno maximálně 50 zaslání s rozestupem půl sekundy. Poté je vlákno uspáno na 30 vteřin a celý proces se opakuje.

6.5. Funkce systému

V následujícím textu jsou popsány veškeré implementované funkce systému.

6.6. Inicializace

Probíhá při startu systému. První věcí, která se provádí je načtení konfiguračního souboru pomocí metody:

```
get_config(CONFIG_FILENAME)
```

při tomto procesu se naplní jednotlivé struktury, které jsou potřebné pro běh systému. Jsou to struktury:

- `struct config_files configfiles;`
- `struct config_pgp configpgp;`
- `struct config configstruct;`

Po tomto procesu se program připojí do databáze pomocí metody:

```
connect_db(configstruct.db_name)
```

Následně se zkontroluje zda je příslušná tabulka FILES obsažena v databázi a je případně vytvořena. Dále se nastaví požadované limity na minimální volnou kapacitu disku. Pokračuje se vytvořením vlákna s File Serverem.

Po těchto operacích následuje postup spuštění Chimery, který je popsán v sekci 6.3. Po spuštění Chimery se vytvoří vlákna modulů pro získávání chunků a mazání chunků.

Pokud server není zakládací vytvoří se vlákno na začlenění do systému. Tato funkce je popsána v sekci 6.13.

Nakonec se spustí nekonečná smyčka pro výpis informací a odpojení ze systému pomocí příkazů z konzole.

6.7. Uložení chunku

Ukládání nového chunku je realizované výzvou AS na některý z datových uzlů DHT sítě na port na, kterém běží File Server. FileServer vytvoří pro novou výzvu nové vlákno, kterému předá atributy spojení. Toto vlákno provede bezpečnostní handshake s AS a přijme jeho požadavek, který rozparsuje a pokud se jedná o kód

`FIND_KEY_FOR_SAFE`

je tento požadavek poslán prostřednictvím Chimery:

```
chimera_send (state, key, code, LENGTH, message);
```

a jako klíč pro směrování je použito prvních 40bajtů z identifikátoru chunku.

Požadavek díky Chimeře dojde k správnému adresátovi a je zpracován přes callback funkci, která na základě kódu zprávy vytvoří nové vlákno, které pracuje s metodou

```
void * ini_cli_safe (void *parameters)
```

a předá mu přijatý požadavek.

Nové vlákno provede rozparsování požadavku. Poté se provede sestup do stromové struktury složek kam má být chunk uložen. Zkontroluje se zda už v příslušné složce není chunk se stejným identifikátorem. Pokud ano je AS tato informace sdělena pomocí informační zprávy

`FIND_KEY_FOR_SAFE:1:NAME`

a proces ukládání končí. Pokud soubor není obsažen, ale není na disku dostatečně volná kapacita pro uložení chunku je opět o tomto informován AS zprávou:

`FIND_KEY_FOR_SAFE:2:NAME`

a operace je ukončena. Pokud je naopak vše v pořádku provede se odeslání zprávy

`FIND_KEY_FOR_SAFE:0:NAME`

Poté po obdržení potvrzovací zprávy je zahájen vlastní přenos chunku. Pokud nastane během přenosu chyba je celý proces anulován a přijatá data jsou vymazána. Pokud je soubor úspěšně přijat a uložen je vytvořen nový záznam do databázové tabulky FILES a následně odeslána potvrzovací zpráva operace. Poté je komunikace s AS ukončena.

Dále se provede analýza zda v úložišti není už obsažen tento soubor. Provede se na základě hledání v záznamech FILES zda existují dva a více stejných záznamů se stejným KEY identifikátorem. Pokud je už takovýto soubor obsažen, je nově vytvořený chunk smazán a vytvořen symlink na starší záznam. Po této operaci pokud je u chunku nastaveno počet replikací více jak 0 provedena replikace chunku, které je popsáno v sekci 6.11.

6.8. Čtení chunku

Zaslání chunku je realizované výzvou AS na některý z datových center DHT sítě na port na kterém běží File Server. File Server vytvoří nové vlákno kterému předá atributy spojení. Toto vlákno provede bezpečnostní handshake s AS a přijme jeho požadavek, který rozparsuje a pokud se jedná o kód

```
FIND_KEY_FOR_SEND
```

je tento požadavek poslán prostřednictvím DHT sítě rovněž pod stejným kódem a jako klíč je použito prvních 40bajtů z identifikátoru chunku.

```
chimera_send (state, key, code, LENGTH, message);
```

Požadavek díky Chimere dojde k správnému adresátovi a je zpracován přes callback funkci, která na základě kódu zprávy vytvoří nové vlákno, které pracuje s metodou

```
void * ini_cli_read (void *parameters)
```

a předá mu přijatý požadavek.

Nové vlákno provede rozparsování požadavku. Poté se provede sestup do stromové struktury složek, kde má být čtený chunk uložen. Pokud zde není nalezen je nahlédnuto do databázové tabulky FILES zda existuje záznam s identifikátorem chunku. Pokud není nalezen je AS informován, že takovýto chunk se v datovém centru nenachází zprávou

```
FIND_KEY_FOR_SEND:NAME:SIZE:1
```

. Pokud ano je dotaz přeposlán nejbližšímu pravému sousedovi z tabulky pravých sousedů a toto vlákno je ukončeno. Důvodem je pokus o vyhledání replikace chunku. Pokud počet takovýchto přeposlání překročí velikost tabulky sousedů, je AS rovněž odeslána zpráva o nedostupnosti požadovaného chunku.

Pokud je soubor nalezen provede se odeslání informační zprávy:

```
FIND_KEY_FOR_SEND:NAME:SIZE:0
```

k AS. Pokud AS potvrdí operaci pomocí potvrzovací zprávy, je spuštěno odeslání souboru. Po úspěšném odeslání se ještě počká na potvrzující zprávu od AS a poté je aktualizován záznam časový záznam odeslaného chunku v tabulce FILES na hodnotu aktuálního času.

Po těchto operacích činnost vlákna končí a je ukončeno.

6.9. Aktualizace časového záznamu chunku

Aktualizací časových záznamu se provádí na výzvu, kterou vznese AS na některý z datových center DHT sítě na port na kterém běží File Server. File Server vytvoří nové vlákno kterému předá atributy spojení. Toto vlákno opět provede bezpečnostní handshake s AS a přijme požadavek, který rozparsuje a pokud se jedná o kód

```
UPDATE_CHUNKS
```

je tento požadavek dále zpracováván. Provede se odeslání potvrzovací zprávy k AS. Následně se provede přijmutí souboru s identifikátory chunků, u kterých má být zaktualizován časový záznam. Poté se provede vložení všech přijatých záznamů do databáze, konkrétně do nově vzniklé tabulky ve formátu podle obrázku [9], která má jméno podle identifikátoru SEQ z požadavku AS. Následně je odeslána potvrzovací zpráva AS a je s ním komunikace ukončena. Následuje proces kdy jsou jednotlivé identifikátory čteny z databáze po 25 záznamech a jsou posílány prostřednictvím DHT sítě k datovým centrům, které tyto chunky vlastní zprávou:

```
chimera_send (state, key, UPDATE_CHUNKS, strlen(sql), sql);
```

poté se vždy hodnata SEND u poslaného identifikátoru inkrementuje. Po odeslání těchto 25 záznámů je vlákno uspáno na 20 sekund a poté se činnost opakuje.

Tento proces je prováděn to té doby než se tabulka nevyprázdní, nebo v tabulce není identifikátor s záznamem SEND menší než 2.

Uzly které přijmou prostřednictvím Chimery přes callback funkci požadavek na aktualizaci časového záznamu vytvoří vlákno, které pracuje s metodou

```
void * ini_cli_update (void *parameters)
```

Tato metoda provede rozparsování požadavku a provede aktualizaci časového záznamu chunku. Poté se spojí s datovým centrem který tento požadavek vznesl a odešle mu zprávu

```
UPDATED_CHUNKS:SEQ:NAME
```

Poté komunikaci ukončí. Následně pokud záznam obsahuje repliky, odešle požadavek na příslušná datová centra o zaktualizování časových záznamů těchto replik. Po této operaci činnost vlákna končí.

Datové centrum, které přijme zprávu

```
UPDATED_CHUNKS
```

prostřednictvím File serveru, rozparsuje tuto zprávu. Následně provede vymazání příslušného chunku z tabulky pod identifikátorem SEQ.

6.10. Mazání chunku (periodické)

Představuje samostatný modul, který je spouštěn při inicializaci serveru.Vlastní činnost modulu je v samostatném vlákně, které pracuje s metodou

```
void * delete_my_old_files (void *parameters)
```

Samotné vlákno se při incializaci se ihned uspí na 24h. Poté se provede pomocí SQL příkazů načtení všech záznamů z tabulky FILES, které mají čas TIME starší více jak 90dní. Každý tento záznam je poté použit k nalezení příslušného souboru na disku. Poté je tento soubor smazán a následně i záznam z SQL tabulky FILES. Po zpracování veškerých záznamů je vlákno opět uspáno na 24h a cyklus se opakuje.

6.11. Replikace chunku

Replikace chunku začíná v případe uložení nového chunku jak bylo popsano v sekci 6.7. Vlákno pro uložení chunku odešle na základě počtu replikací jednotlivým datovým centrům z tabulky pravých sousedů informační zprávu s kódem pro požadavek o uložení repliky

```
chimera_send (state, send_key, REP_SAFE, LENGTH ,revbuf);
```

poté činnost tohoto ukládacího vlákna končí.

Příslušné uzly dostanou prostřednictvím Chimery tyto požadavky a zpracují je pomocí callback funkce. Vytvoří nové vlákno, které používá stejnou metodu jako pro ukládání, ale s kodem pro uložení repliky, tedy:

```
void * ini_cli_safe (void *parameters)
```

Další činnost je tedy stejná jako v případě samotného ukládání chunku s drobnými úpravami. Například při vkládání chunku do tabulky FILES v databázi je hodnota UID nastavena na uzel, který tuto činnost vyvolal a také není proveden proces replikace.

Toto vlákno které tedy ukládá repliku se připojují k File Serveru uzlu, který chunk vlastní pomocí zaslání kódu o uložení replikace. File server v novém vlákně požadavek zpracuje a obslouží dotazované datové centrum posláním chunku, který je vyžadován.

6.12. Výpadek datového centra

Pokud nastane výpadek nějakého datového centra jsou ostatní datová centra, která vlastní v tabulce sousedů toto vypadnuté datové centrum informování pomocí callback funkce

```
void update_message (Key * k, ChimeraHost * h, int joined)
```

Uzly si tedy opraví tabulku nejbližších pravých a levých sousedů.

Díky této operaci zjistí na jaké pozici v levé a pravé tabulce se naházel. Díky této informací je možné začít s nápravou počtu replikací a nápravou identifikátorů vypadlého datového centra.

Tato činnost vypadá následovně. Pokud vypadlé datové centrum bylo můj nejbližší levý soused. Znamená to že převezmu část jeho stavového prostoru. Tudíž se spustí metoda

```
int update_to_lead(char key[])
```

která u všech záznamů v databázové tabulce FILES s UID vypadnutého datového centra na staví UID na prázdnou hodnotu. Touto operací si příslušný chunk přivlastní. Po skončení této operace je zaslána informační zpráva prostřednictví DHT sítě

```
chimera_send(state, *leftleafkeys[0],CAN_UPDATE,strlen(buffer)+1,buffer);)
```

novému nejbližšímu levému sousedu. Poté se provede rozeslání na ostatní levé sousedy požadavek o získání nových identifikátorů replik, které nyní na tento uzel připadají.

```
chimera_send (state,*leftleafkeys[i],UP_REP, strlen (buffer)+1,buffer);
```

Ostatní dotčené uzly provedou změnu v databázové tabulce FILES u všech záznamů s UID vypadlého na UID toho uzlu, který jej v tabulce nahradil. Poté se provede rozeslání na ostatní levé uzly požadavek o získání nových identifikátorů replik které nyní na tento uzel připadají.

Uzel, který obdrží informační zprávu

```
CAN_UPDATE
```

provede pomoci metody

```
int get_my_record ()
```

zaslání požadavku na odeslání všech identifikátorů, které mu nyní po výpadku datového centra připadají pomocí zprávy s kódem:

```
GET_MY_RECORD
```

File Servery datových center, které přijmou požadavek

```
GET_MY_RECORD nebo UP_REP
```

Zpracují tyto požadavky prostřednictvím metody:

```
void * send_my_rep (void *parameters)
```

Pokud je požadavek formátu

```
GET_MY_RECORD
```

jsou vybrány všechny záznamy z tabulky FILES v databázi, které nemají vyplněné UID. Následně jsou přefiltrovány algoritmem, který z nich vybere pouze ty, které spadají do stavového prostoru uzlu, který tento požadavek vznesl. Poté jsou tyto záznamy uloženy do souboru. Následuje připojení k datovému centru, který tyto identifikátory požadoval společně s odesláním souboru s identifikátory chunků. Pokud se jedná o případ

```
UP_REP
```

, jsou z databázové tabulky FILES vybrány všechny identifikátory chunku s příslušným číslem replikace o které je žádáno. Následuje připojení k datovému centru, který tyto identifikátory požadoval společně s odesláním souboru s identifikátory chunků.

Datové centrum, které prostřednictvím File Serveru obdrží zprávy

```
GET_MY_RECORD nebo UP_REP
```

Je zpracují metodou:

```
int received_new_rep(gnutls_session_t session,int new_sock,char key[],
int code,double size)
```

Metoda příslušné identifikátory chunků uložené v souboru přijme a následně je nahraje do tabulky FILES s příznakem FLAG 0 a UID, které označuje kde mají být chunky uloženy. Tím začleňování končí a přejímá iniciativu modul získávání chunků, které je popsáno v sekci 6.4.4.

6.13. Připojování datového centra

Připojování datového centra je realizováno postupně. Samotné začlenění do DHT nám zajistí Chimera, na nás tedy zbývá zrealizovat načtení všech chunků, které mají být nově na tomto uzlu obsaženy.
Při incializaci na připojovaném se tedy vytvoří vlákno, které provede následující. Odešle požadavek na svoje nejbližší levé a pravé datové centrum o zaslání identifikátorů chunků, které nyní spadají do jeho stavového prostoru prostřednictvím zprávy přes DHT síť

```
chimera_send(state,*globrightleafkeys[0],GET_MY_RECORD,strlen(buffer)+1,buffer);
```

následně poté se spustí činnost, která si zažádá o všechny identifikátory replik, které má nyní tento uzel vlastnit. To znamená, že svému nejbližšímu uzlu pošle požadavek o repliky s číslem replikace 1 až 4 pomocí DHT sítě s kódem zprávy o zaslání identifikátorů

```
chimera_send (state, *globleftleafkeys[i], UP_REP, strlen (buffer) + 1,buffer);
```

Po této činnosti funkce vlákna končí a je ukončeno.

File Servery datových center, které přijmou požadavek

```
GET_MY_RECORD nebo UP_REP
```

Zpracují tyto požadavky prostřednictvím metody:

```
void * send_my_rep (void *parameters)
```

Pokud je požadavek formátu

```
GET_MY_RECORD
```

jsou vybrány všechny záznamy z tabulky FILES v databázi, které nemají vyplněné UID. Následně jsou přefiltrovány algoritmem, který z nich vybere pouze ty, které spadají do stavového prostoru uzlu, který tento požadavek vznesl. Poté jsou tyto záznamy uloženy do souboru. Následuje připojení k datovému centru, který tyto identifikátory požadoval společně s odesláním souboru s identifikátory chunků. Pokud se jedná o případ

```
UP_REP
```

, jsou z databázové tabulky FILES vybrány všechny identifikátory chunku s příslušným číslem replikace o které je žádáno. Následuje připojení k datovému centru, který tyto identifikátory požadoval společně s odesláním souboru s identifikátory chunků.

Náš připojovaný uzel prostřednictvím File Serveru obdrží zprávy

```
GET_MY_RECORD a UP_REP
```

a zpracuje je metodou:

```
int received_new_rep(gnutls_session_t session,int new_sock,char key[]
,int code,double size)
```

Metoda příslušné identifikátory chunků uložené v souboru přijme a následně je nahraje do tabulky FILES s příznakem FLAG 0 a UID, které označuje kde mají být chunky uloženy. Tím začleňování končí a přejímá iniciativu modul získávání chunků, které je popsáno v sekci 6.4.4.

6.14. Přepsání chunku

Přepsání chunku probíhá stejně jako ukládání chunku s drobnými změnami. Identifikační kód je:

`FIND_KEY_FOR_RSAFE`

6.15. Přepsání repliky chunku

Přepsání repliky chunku začíná v případě přepisu chunku jak bylo popsáno v sekci 6.14. Proces je takřka stejný s replikací chunku až na drobné změny. Kdy identifikační kód pro přepis je:

`REP_RSAFE`

6.16. Monitoring

Jednotlivé monitorovací atributy jsou řešeny následovně:

- Počet klientských stanic: řešeno pomocí sdílené proměnné
  `count_client`
  , která je inkrementována vždy při navázání spojení s klientskou stanicí na File Serveru. Následně je dekrementována po odpojení klientské stanice.
- Počet serverových spojení: jedná se o sdílenou proměnou
  `count_server`
  , která představuje počet aktivních spojení s AS, nebo jinými datovými centry, kdy uzel se připojuje jako klient. Hodnota je ikrementována vždy na začátku spojení a dekrementována po ukončení spojení.
- Celková velikost místa na disku: je realizovaná pomocí metody:
  `char * get_capacity ( char * dev_path)`
  a vrací hodnotu v MB
- Celkové volné místo na disku: je realizované pomocí metody:
  `char * get_free_space ( char * dev_path)`
  a vrací hodnotu v MB

7. Testování

V této kapitole se budeme zabývat testováním pilotní implementace distribuovaného datového centra a testováním v rámci celého systému distribuovaného uložiště.

7.1. Testování v rámci subsystému

Testování mého subsystému probíhalo převážně manuálním způsobem, simulováním různých situací. V počátku bylo nutné vyvinout jednoduchý AS, který následně posloužil jako testovací rozhraní pro různé operace. Během tohoto testování vyplinuly některé nedostatky knihovny Chimera, které byly popsány v sekci Realizace 6.2. Při tomto testování jsem se zaměřoval na testování stability a spolehlivosti systému.

Testování probíhalo zpočátku pouze na localhostu se dvěmi až šesti datovými centry, následně se přešlo na testování na dvou počítačích zapojených v LAN síti.

Při tomto testování bylo testováné celé spektrum implementovaných funkcí. Jak od základních úloh, načtení dat, uložení dat, aktualizaci časových záznamů. Tak po funkce zajišťující stabilitu a škálovatelnost jako je odpojení uzlu a připojení uzlu. V případě replik bylo testováno až do urovně 4 replikací.

Během testování bylo také ověřeno že TCP spojení je zabezpečeno. K tomuto posloužil program WireShark, kterým bylo možno odchytit celou komunikaci. Z výsledných dat nebylo možné získat jakkékoliv informace.

Dále bylo během testování také sledováno celková zatěž na systém pomocí "sledování systémových prostředků", který je implicitně integrován do systému Ubuntu.

Nakonec jsem i při testování používal nástroje jako je valgrind a gdb, který byl v odhalování chyb velice prospěšný.

7.2. Testování v rámci celého systému

Testování v rámci celého systému zpočátku probýhalo na zakládní funkce s alfa verzemi. Kdy byli zřízena dvě datová centra, která zajišťovala pouze základní funkcionalitu na mém osobním počítači, která sloužila pro Bc. Jana Januru jako počátěční testovací rozhraní při vývoji a testování jeho AS. Nakonec se přešlo na testováni finálních verzí, kdy se testovala především základní funkcionalita.

7.2.1. Testovací sestava

Jednotlivá datová centra a jeden access server byly umístěny na pět virtuálních serverů umístěných v síti ČVUT FEL na Karlově Náměstí. Jejich paremtry byly následující:

- 1xVCPU,0,5GB RAM, 20GB HDD Debian 7.0.3 64b Všechny tyto virtuální stroje vlastnily veřejnou IP.

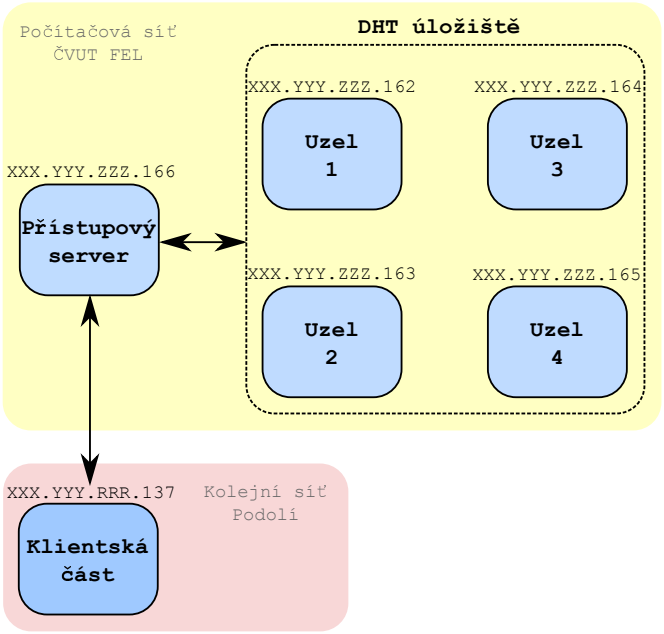
Vlastní konfigurace serveru na kterém tyto virtuální stroje běží je následující:

- Hardware - Fujitsu Primergy RX300
  - Procesor - 2 x Intel(R) Xeon(R) CPU E5-2620 v2 @ 2.10GHz
  - Operační paměť - 128 GB DDR3/RDIMM 1600
  - Síťové připojení - 2 x 10 GbE
  - HDD - 4 x 600 GB 3.5"@ 15k SAS (RAID 5)SW
- Software
- Linux cn01 3.2.0-4-amd64 1 SMP Debian 3.2.51-1 x86_64 GNU/Linux
- QEMU emulator version 1.1.2 (qemu-kvm-1.1.2+dfsg-6, Debian), Copyright (c) 2003-2008 Fabrice Bellard
- libvirtd (libvirt) 0.9.12.3

Klientská část běžela na osobním počítači na koleji Podolí s rychlostí připojení 1Gb/s symetricky. Parametry klientského počítače byly následující:

- Procesor: Intel(R) Core(TM)2Duo CPU T6500 2,1GHz
- RAM: 4GB DDR2
- HDD: 20GB
- Operační systém: Ubuntu 12.04LTS 64bit

Célé testovací schéma je zobrazeno na Obr. 10.



Obrázek 10. Schéma testovací konfigurace

7.2.2. Výsledky měření

Testování proběhlo úspěšně na malé soubory. Byly úspěšné zapsány a následně všechny úspěšně přečteny. U všech souborů byla nastavena replikace na jednu repliku, která se úspěšně vytvářela. Také jsme si měřením ověřili, že bezpečnostní handshake velice značně zpomaluje celou komunikace. Dále to bylo zapříčiněno sekvenčním zpracováváním na straně klienta, kdy se další operace vykonávala až po úspěšném provedení předchozí. Naměřené výsledky stopovaných časů nahrávání byly následující:

- Textový soubor 250kB - 12s
- Fotografie 1.2MB - 13s
- Zvukový soubor 6.3MB - 15s
- Soubor ve formátu ZIP 10.9MB - 15s

Při testu stahování, byla stahována celá struktura uživatelovo složky, která vypadala následovně:

```
root
├── file.txt
├── directory1
│   ├── fotka1.jpg
│   └── fotka2.jpg
└── directory2
    ├── song.mp3
    └── file.zip
```

tato celá struktura o velikosti 20MB byla po zapnutí klienta celá po procházení dostupná za 35s.

Během testování se na klientské aplikaci objevil problém s nahráváním větších souborů, nebo většího množství malých souborů najednou (např. zkopírováním složky) při kterém docházelo k nekorektnímu ukončování aplikace.Tento problém byl způsobený špatnou detekcí ukončení zápisu do souboru, který se projevil až při závěrečném testování.

Avšak díky tomuto bylo možné datová centra otestovat na spolehlivost. Jednotlivým datovým centrum byly takto zasílány zprávy ve špatném formátu, špatnými parametry, nebo při výměně souborů bylo spojení ukončeno. Datová centra nicméně zůstala provozu schopná a provádělo následní operace korektně.

Problém byl poté odstraněn a tak mohlo dojít i k otestování velkých, nebo mnoha souborů. Výsledky stopovaných časů byly následující:

- Složkka 12 fotografií o celkové velikosti 23.4MB - 5m (0,62Mb/s)
- Sobor ve formátu ZIP 23.1MB - 1m (3Mb/s)
- Soubor ve formátu ZIP 51.3MB - 1m30s (4,56Mb/s)
- 150 fotografií zabalené ve formátu ZIP 335.4MB - 4m (11,18Mb/s)

tato celá struktura o velikosti 434,3MB byla po zapnutí klienta celá po procházení dostupná za 7m30s (7,72Mb/s).

Z měření se tedy ukázalo, že v případě mnoha souborů je dosažená rychlost velice pomalá, díky zpracovávání na klientské stanici. Naopak čím byl soubor větší, tím celková přenosová rychlost narůstala, díky nižší režii na klientské části.

Podařilo se nám tedy otestovat celý systém dohromady, bohužel se ukázali některé implementační nedostatky a zjednodušení částí systému. Naměřené časy jsou také velice pomalé a je nutné zoptimalizovat vlastní bezpečnostní handshake a práci na klientské části. Pilotní implementace nicméně už nyní může sloužit jako jednoduché úložiště na data.

8. Závěr

Tato práce se zabývala subsystémem distribuovaného úložiště. Konkrétně subsystémem pro vlastní uložení dat. Na začátku práce jsem vypracoval rešerši části nejznámějších služeb pro distribuované ukládání dat s jejich následným otestováním. Poté jsme provedl lehký úvod do celkového projektu distribuovaného úložiště. Následně byly poznatky z rešerše použity při analýze a návrhu subsystému distribuovaného datového centra. Následovala realizace vlastního distribuovaného datového centra, které bylo poté otestováno.

Výsledkem je tedy pilotní implementace, která splnila zadání. Jsou zde obsaženy funkce pro ukládání, čtení, replikaci, začleňování nových datových center a ošetření výpadků datových center. Bohužel se nepovedlo naimplementovat veškeré funkce zmíněné v analýze. Systém, konkrétně DHT není plně zabezpečen. Tudíž není chráněna například na podvržení zpráv. Dále systém díky, nenaimplementování funkcí pro promazání replik, bude obsahovat více redundantních dat.

Během psaní práce jsem se nejvíce potýkal s problémy okolo knihovny Chimera. Tato knihovna se zpočátku jevila jako dobře použitelná, nicméně s postupem času se ukázala jako nepříliš povedená. Bohužel v době tvorby nebyla k dispozici žádná jiná dobře zdokumnetovaná knihovna s vhodnou licencí. Proto pro budoucí rozvoj bych doporučoval vytvoření vlastní DHT knihovny, což samo o sobě je však velmi komplikovaná úloha.

Další rozvoj práce je tedy v DHT knihovně a to buď v komplexních úpravách, nebo v již zmíněném doporučení o vlastní knihovně. Dále v dodělání nenaimplementovaných funkcí. Poté se může další rozvoj práce ubírat mnoha směry, například v optimalizaci.

Celkově tato práce mi hodně dala a vzala. Prohloubil jsem si znalosti v programovacím jazyku C. Naučil jsem se používat některé nové knihovny. Mohl jsem využít mnoho znalosti jako například z operačních systému, nebo distribuovaných systémů. Vyzkoušel si pracovat v týmu s Bc. Janem Janurou, který měl na starost AS a klientskou část a Bc. Tomášem Dyntarem, který měl nestarost správu metadat. Vlastní zkušeností jsem si ověřil, že velké projekty vyžadují velké úsilí.

Příloha A.

Instalační a uživatelská příručka

A.1. Prostředí pro běh

Systém byl vyvíjen a testován na operačním systému Ubuntu 12.04LTS 32bit a testován na systému Debian 7.0.3 64bit.

Program by měl být použitelný i na platformě Windows. Jednotlivé použité knihovny jsou kompatibilní se systémy Windows. Jen v případě použitých POSIX vláken je nutné doinstalovat knihovnu Pthreads-w32 pro 32bit systémy, nebo případně Pthread-winx64 pro 64bit systémy.

Systém pro svůj chod používá používá tyto knihovny:

1. GnuTLS
2. SQLite

A.2. Kompilace

Před kompilací je nutné doinstalovat některé další balíčky. Jejich seznam je následovný:

1. M4
2. GMP
3. Nettle
4. GnuTLS 3.1.10 a novější
5. libgnutls-dev
6. libsqlite3-dev
7. OpenSSL (požaduje Chimera SHA-1)

8. Chimera 1.20A

Knihovna Chimera 1.20A je upravenou knihovnou Chimera 1.20 a je přibalena k programu. Její instalace je standartní:

```
$./configure
$make
$make install
```

Implementace datového centra obsahuje soubor makefile. Díky tomu je kompilace jednoduchá – stačí použít příkaz

```
$make
```

A.3. Nastavení

Konfigurce každého datového serveru je přez konfigurační soubor. Jeho názvem musí být config.conf. Jeho obsah je následující:

```
***CONFIGURE FILE OF DISTRIBUTED DATA CENTER***
SERVER=0
NAMESERVER=DCentr1
DHTPORT=5000
DHTPINGPORT=5001
FILESERVERPORT=5003
IPSERVER=
PORTSERVER=0
DATADIR=data
DATABASE=filDB.db
DISKPATH=/
CACHEDIR=/cache/
DIRECTORIES=1
KEYFILE=secret.asc
RINGFILE=ring.gpg
CERTFILE=public.asc
***CONFIGURE FILE OF DISTRIBUTED DATA CENTER***
```

Řádek SERVER značí zda datové centrum zakládá novou dht síť s datovými uzly a to hodnotou 0. V případě hodnoty 1 se spuštěné datové centrum pokusí připojit do již existující DHT sítě, pomocí jednoho známého uzlu identifikovaného pomocí řádků IPSERVER a PORTSERVER.

Řádek NAMESERVER označuje jméno datového centra se a slouží jako základ pro vytvoření hashe kterým je uzel v DHT síti identifikován.

Řádky DHTPORT, DHTPINGPORT a FILEPORT identifikují na jakých portech naslouchá směrování DHT sítě Chimera respektive naslouchá DHT proces dotazování o životě uzlů z směrovacích tabulek respektive naslouchá server pro výměnu souborů mezi access serverem, nebo jinými datovými uzly.

Řádek DATABASE označuje název souborů databáze, který se má použít. Řádky CACHEDIR a DATADIR označují název složky, kde budou uloženy dočasné soubory respektive samotné chunky dat. Řádky KEYFILE, RINGFILE a CERTFILE určují názvy použitého soukromého klíče, ringfilu a veřejného klíče.

A.4. Spuštění

Jednotlivé datová centra je nutné spouštět v správném pořadí. Tedy nejdříve spustit datové centrum které vytvoří DHT síť a poté ostatní uzly, které se k DHT síti připojují. Po uspěšném spuštení se zobrazí následující řádky:

```
Opened database successfully
File Server ready. Listening to port '5003'.

Data Center name:DCentr1 key:2b8158c8675214e5c1edfb0d6fa4d691018fa4b4
DHT port:5000 DHT port ping:5001
Communication between Data Storage <key> <message> **

Name Data Storage: OR Command help
```

A.5. Příkazy za běhu programu

V běžícím programu lze využít některé příkazy pro zobrazení různých informací. Jejich seznam je následující:

```
$help
$SHFC
$SHCA
$SHCC
$SHSC
$LFD
```

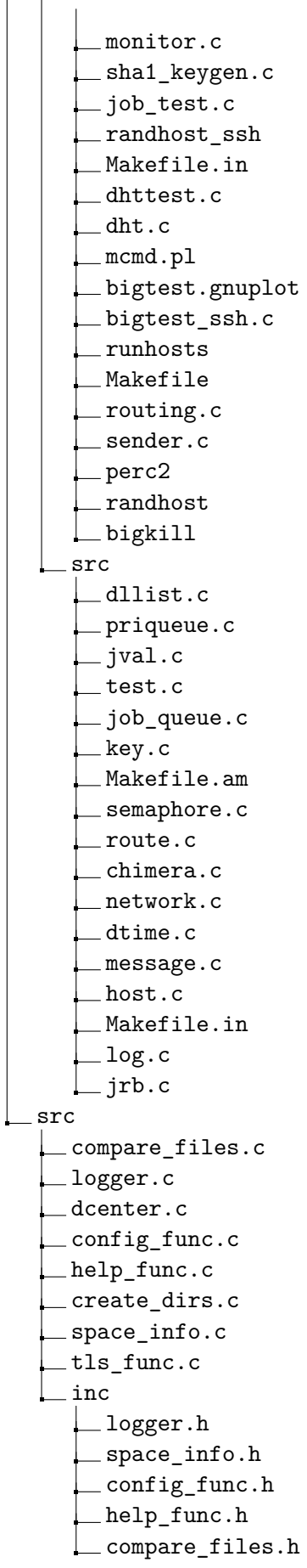
Příkaz help zobrazí všechny možné příkazy s celým názvem.
SHFC zobrazí aktuální volnou kapacitu na disku v MB.
SHCA zobrazí aktuální celkovou kapacitu disku. SHCC zobrazí počet všech aktuálně připojených klientů.
SHSC zobrazí počet aktivních spojeních které aktuálně datové centra má.
Příkaz LFD provede násilné ukončení datové centra.

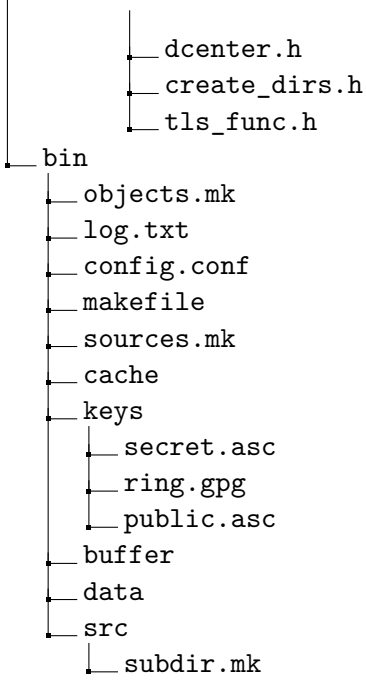
Příloha B.

Obsah přiloženého CD

```
/
├── text
│   ├── abbreviations.tex
│   ├── abstract.tex
│   ├── acknowledgement.tex
│   ├── app01.tex
│   ├── app02.tex
│   ├── archi.pdf
│   ├── cp.tex
│   ├── declaration.tex
│   ├── felthesis.cls
│   ├── ch01.tex
│   ├── ch02.tex
│   ├── ch03.tex
│   ├── ch04.tex
│   ├── ch05.tex
│   ├── ch06.tex
│   ├── ch07.tex
│   ├── ch08.tex
│   ├── chimeraarchi.pdf
│   ├── kudrnmar.bib
│   ├── kudrnmar.pdf
│   ├── kudrnmar.tex
│   ├── lev.pdf
│   ├── small-probabilities.pdf
│   ├── spider1.pdf
│   ├── spider2.pdf
│   ├── spolecnatabulka.jpg
│   ├── spolecnynavrh.pdf
│   ├── sqltable.pdf
│   ├── table2.pdf
│   └── testovani.pdf
└── source
    └── chimera
        ├── install-sh
        ├── COPYING
        ├── libtool
        ├── missing
        ├── README
        └── Makefile.am
```

```
├── config.guess
├── config.log
├── config.status
├── ChangeLog
├── bootstrap
├── AUTHORS
├── acinclude.m4
├── NEWS
├── mkinstalldirs
├── Makefile.in
├── CHANGES
├── configure
├── INSTALL
├── Makefile
├── configure.in
├── config.sub
├── ltmain.sh
├── aclocal.m4
├── include
│   ├── semaphore.h
│   ├── priqueue.h
│   ├── network.h
│   ├── log.h
│   ├── jval.h
│   ├── key.h
│   ├── job_queue.h
│   ├── base.h
│   ├── include.h
│   ├── dtime.h
│   ├── route.h
│   ├── chimera.h
│   ├── message.h
│   ├── jrb.h
│   ├── host.h
│   └── dllist.h
└── test
    ├── readme.txt
    ├── keygen.c
    ├── chat.c
    ├── dhttest.c
    ├── mon.gnuplot
    ├── bighost.c
    ├── test.c
    ├── dht.h
    ├── Makefile.am
    ├── receiver.c
    ├── bigtest.c
    ├── bignode.c
    ├── graphmonkey
    └── style.gnu
```


monitor.c
sha1_keygen.c
job_test.c
randhost_ssh
Makefile.in
dhttest.c
dht.c
mcmd.pl
bigtest.gnuplot
bigtest_ssh.c
runhosts
Makefile
routing.c
sender.c
perc2
randhost
bigkill
src
dllist.c
priqueue.c
jval.c
test.c
job_queue.c
key.c
Makefile.am
semaphore.c
route.c
chimera.c
network.c
dtime.c
message.c
host.c
Makefile.in
log.c
jrb.c
src
compare_files.c
logger.c
dcenter.c
config_func.c
help_func.c
create_dirs.c
space_info.c
tls_func.c
inc
logger.h
space_info.h
config_func.h
help_func.h
compare_files.h


dcenter.h
create_dirs.h
tls_func.h
bin
objects.mk
log.txt
config.conf
makefile
sources.mk
cache
keys
secret.asc
ring.gpg
public.asc
buffer
data
src
subdir.mk

Literatura

[1] Jan Janura. *Distribuované úložiště dat - klientská část*. Diplomová práce. Praha, 2014.

[2] Tomáš Dyntar. *Distribuované úložiště dat - správa metadat*. Diplomová práce. Praha, 2014.

[3] Jan Janeček. *Distribuované systémy*. Vyd. 1. Praha: Vydavatelství ČVUT, 2001.

[4] *PowerEdge R710 Rack Server*. Dell Inc. URL: `http://www.dell.com/us/business/p/poweredge-r710/pd` (cit. 02. 02. 2014).

[5] *CLARiiON AX4 Easy-to-use, affordable networked storage*. EMC Corporation. URL: `http://www.emc.com/products/detail/hardware/clariion-ax4.htm` (cit. 02. 02. 2014).

[6] *PhotoDNA*. Microsoft Corporation. URL: `https://www.microsoft.com/en-us/news/presskits/photodna/` (cit. 02. 02. 2014).

[7] *SkyShellEX*. URL: `http://ssx.codeplex.com/` (cit. 02. 02. 2014).

[8] *Wine*. Wine Team. URL: `http://www.winehq.org/` (cit. 02. 02. 2014).

[9] *AllShare Play*. Samsung Group. URL: `http://www.samsung.com/cz/allshare/play/main.html` (cit. 02. 02. 2014).

[10] *Amazon S3*. Amazon.com, Inc. URL: `http://aws.amazon.com/s3/` (cit. 02. 02. 2014).

[11] *Wuala - Web Access*. Wuala. URL: `http://www.wuala.com/en/launch/` (cit. 02. 02. 2014).

[12] *SpiderOak | Online File Sharing Cloud Backup Software | Private Secure Data Storage for Business Home*. SpiderOak. URL: `http://spideroak.com` (cit. 02. 02. 2014).

[13] *MaidSafe - The New Decentralized Internet*. MaidSafe. URL: `http://maidsafe.net/` (cit. 02. 02. 2014).

[14] *OpenP2P*. URL: `http://openp2p.org/Main_Page` (cit. 02. 02. 2014).

[15] Jeremie Miller. *TeleHash / JSON + UDP + DHT = Freedom*. URL: `http://www.telehash.org/` (cit. 02. 02. 2014).

[16] *Chimera*. CURRENT Lab, U. C. Santa Barbara. URL: `http://current.cs.ucsb.edu/projects/chimera/index.html` (cit. 03. 02. 2014).

[17] Petr Tuček. *Distribuované úložiště dat*. Diplomová práce. Praha, 2012. URL: `https://dip.felk.cvut.cz/browse/pdfcache/tucekpe2_2012dipl.pdf`.

[18] *Hash Collision Probabilitiesí*. Jeff Preshing. 2011. URL: `http://preshing.com/20110504/hash-collision-probabilities/` (cit. 02. 02. 2014).

[19] Simon Josefsson Nikos Mavrogiannopoulos. *The GnuTLS Transport Layer Security Library*. GnuTLS. URL: `http://gnutls.org/l` (cit. 03. 02. 2014).

[20] *SQLite*. URL: `https://sqlite.org/` (cit. 02. 02. 2014).

[21] Martin Volf. *Distribuované úložiště dat*. Diplomová práce. Praha, 2014. URL: https://support.dce.felk.cvut.cz/mediawiki/images/6/69/Dp_2014_volf_martin.pdf.

[22] George F Coulouris. *Distributed systems*. 5th ed. Boston: Addison-Wesley, c2012.

[23] *Wuala - Secure Cloud Storage*. Wuala. URL: http://www.wuala.com/ (cit. 02. 02. 2014).

[24] *SugarSync*. SugarSync, Inc. URL: https://www.sugarsync.com/ (cit. 02. 02. 2014).

[25] *Microsoft OneDrive*. Microsoft Corporation. URL: https://onedrive.live.com/ (cit. 02. 02. 2014).

[26] *SugarSync se dočkal řady novinek, odradit však mohou poměrně vysoké ceny*. Internet Info, s.r.o. 2013. URL: http://www.lupa.cz/clanky/sugarsync-prineslo-radu-novinek-odradit-vsak-mohou-pomerne-vysoke-ceny/ (cit. 02. 02. 2014).

[27] *Úložiště dat na internetu: k datům odkudkoli*. oXy Online s.r.o. 2013. URL: http://www.svethardware.cz/uloziste-dat-na-internetu-k-datum-odkudkoli/36307 (cit. 02. 02. 2014).

[28] Mgr. Miroslav Novotný. *Peer-to-Peersítě*. KSI MFF CUNI. URL: ftp://ulita.ms.mff.cuni.cz/predn/PDS/DHT.pdf (cit. 02. 02. 2014).

[29] Pavel Krejsa. *Distibuovaný souborový systém*. Diplomová práce. Praha, 2010. URL: https://dip.felk.cvut.cz/browse/pdfcache/krejspav_2010dipl.pdf.

[30] Michael Welzl. *Peer-to-Peer Systems*. Institute of Computer Science University of Innsbruck,Austria. URL: http://heim.ifi.uio.no/michawe/teaching/p2p-ws08/p2p-5-6.pdf (cit. 02. 02. 2014).

[31] Cyril Klimeš. *Distribuované systémy: texty pro distanční studium*. Praha: Ostravská univerzita v Ostravě, Přírodovědecká fakulta Katedra informatiky a počítačů, 2007.

[32] *Zajímavé grafy a peer to peer sítě*. Masarykova Univerzita. URL: http://is.muni.cz/el/1433/podzim2013/PB165/um/11_peer.pdf?lang=cs (cit. 02. 02. 2014).

[33] *Eclipse*. URL: https://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/keplersr2 (cit. 02. 02. 2014).

[34] *Ubuntu*. URL: http://releases.ubuntu.cz/precise/ (cit. 02. 02. 2014).

[35] *SFU*. URL: http://technet.microsoft.com/en-us/library/bb496506.aspx (cit. 02. 02. 2014).

[36] *Pthreads*. Ross Johnson. URL: https://www.sourceware.org/pthreads-win32/ (cit. 02. 02. 2014).

[37] Mgr. Miroslav Novotný. *Peer-to-Peersítě*. KSI MFF CUNI. URL: ftp://ulita.ms.mff.cuni.cz/predn/PDS/DHT.pdf (cit. 02. 02. 2014).