Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Přehled současných distribuovaných souborových systémů

Roman Vohník

Bakalářská práce

2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Roman VOHNÍK**
Osobní číslo:	**I07984**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Přehled současných distribuovaných souborových systémů**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Teoretická část:
Popište v současnosti dostupné distribuované souborové systémy (Coda, Intermezzo, AFS, Lustre, ...). Zjistěte, zda se jedná o dále vyvíjené produkty, jak je to s jejich podporou a celkově je popište a porovnejte.
Implementační část:
Proveďte instalaci co možná největšího počtu DFS a proveďte na nich benchmark.
Vyhodnoťte náročnost instalace a správy jednotlivých DFS, porovnejte jejich výkon.
Vyberte vhodný DFS pro instalaci do učeben FEI.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

***Kolektiv autorů: LINUX dokumentační projekt, Computer Press 1998 (3. aktualizované vydání), ISBN 80-7226-761-2**

***Shah, S.: Administrace systému Linux - podrobný průvodce začínajícího administrátora, Grada Praha 2002, ISBN 80-7169-586-6**

***Přehled distribuovaných file-systémů: http://en.wikipedia.org/wiki/List_of_file_systemsDistributed_file_systems**

*** benchmarking sw: http://www.acnc.com/benchmarks.html**

***Bonnie: http://www.textuality.com/bonnie/**

***Bonnie++: http://www.coker.com.au/bonnie++/**

Vedoucí bakalářské práce: **Mgr. Tomáš Hudec**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. ledna 2010**

Termín odevzdání bakalářské práce: **14. května 2010**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Cegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury. Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

v Pardubicích dne 12.5.2010

Roman Vohník

Poděkování

Chtěl bych poděkovat panu Mgr. Tomáši Hudcovi za zajímavé a hodnotné téma, za podnětné rady a připomínky a za celkové vedení mé práce.

ANOTACE

Tato práce obsahuje přehled v současnosti dostupných distribuovaných souborových systémů, především pro platformu GNU/Linux. Hlavním obsahem je popis jejich funkčnosti, postup instalace a konfigurace. Součástí jsou také výkonové testy těchto systémů.

KLÍČOVÁ SLOVA

distribuované systémy, souborové systémy, GNU/Linux, síť

TITLE

Review of current distributed file systems

ANOTATION

This work contains review of currently available distributed file systems, especially for GNU/Linux platform. Their description, process of installation and configuration is main part and a series of the performance tests follows.

KEY WORDS

distributed systems, file systems, GNU/Linux, network

Obsah

Úvod

Souborové systémy jsou v podstatě mechanismem pro jednotné a uživatelsky jednoduché ukládání dat na nejrůznější datová úložiště jako jsou pevné disky, paměti flash apod. Data jsou ukládána do objektů nazývaných soubory, ve kterých jsou data uložena spolu se svými metadaty, která mimo jiné obsahují například název souboru, časy vytvoření a poslední změny, oprávnění a další atributy. Soubory jsou z důvodu organizace a přehlednosti ukládány do adresářů, což jsou v podstatě jen soubory s atributem adresáře. Celková struktura je pak stromového typu, kde adresáře mohou obsahovat jak soubory, tak další podadresáře.

Distribuované souborové systémy (dále v tomto textu uváděné také pod zkratkou DFS), jsou též označovány jako síťové. A jak již názvy napovídají, jsou to souborové systémy, které jsou dostupné z prostředí počítačové sítě.

Tato práce si klade za úkol provést přehled a popis volně dostupných distribuovaných systémů pro prostředí GNU/Linux, jejich instalaci a také popis konfigurace. Systémy budou v případě úspěšného zprovoznění testovány některými z dostupných aplikací (benchmarků). Pro dosažení dostatečně vypovídajících hodnot budou testy provedeny na několika počítačových sestavách různé konfigurace. Následně budou zhodnoceny jak výsledky testů tak náročnost zprovoznění a následného užívání daných systémů.

1 Distribuované souborové systémy

1.1 Úvod do DFS

Z hlediska uživatelského komfortu se většina systémů snaží zpřístupnit data tak, jako by byla dostupná lokálně. Pro uživatele to může být tak transparentní, že vůbec nepoznají, že při procházení adresářové struktury jsou některé adresáře a soubory v podstatě virtuální a operační systém zajišťuje komunikaci se serverem, kde jsou data fyzicky uložena.

DFS musí ale řešit pod povrchem spoustu problémů, a to hlavně z důvodu, že pracují po síti, která je stále pomalejší než lokální zpracování. Navíc některé implementují řadu pokročilejších funkcí – a tou nejdůležitější je možnost **distribuce dat na více serverů.** To je využíváno z důvodu zvýšení dostupnosti, ochrany, nebo výkonu. Pokud však máme více serverů, je výhodné, aby bylo umístění dat pro klienta transparentní. Tudíž aby nemusel znát například adresu konkrétního serveru, kde jsou uložena právě jeho data. Toto za něj řeší většinou hlavní server.

1.1.1 Vlastnosti

Důležité bývá zabezpečení ukládaných dat. Řeší se otázka autorizace a autentizace (LDAP, Kerberos) nebo šifrování. Šifrovat se může samotný přenos dat po nezabezpečené síti nebo i ukládaná data. Roli tu hraje také zabezpečení pomocí přístupových práv, například pomocí vlastních ACL (access control list), neboť některé systémy neimplementují standardní POSIX ACL.

Dalšími vlastnostmi může být například implementace vyrovnávací paměti (jak u klienta, tak u serveru), možnost pracovat s daty lokálně i při odpojení od sítě (například na notebooku) anebo možnost zamykání souborů.

Stejně jako u klasických souborových systémů je zde možnost přístupu více uživatelů/procesů ke stejným datům a musí se řešit (v tomto případě složitěji) konkurenční čtení a zápis. Z tohoto pohledu můžeme rozdělit servery DFS na stavové a bezstavové. Stavový server si musí uchovávat informace o všech klientech a jejich otevřených souborech. Klient získá ukazatel na soubor a pomocí něj se dále dotazuje. Oproti tomu bezstavový server neuchovává informace, tudíž není možné zamykání souborů a pro každý dotaz se musí ma-

povat virtuální cesta na ukazatel na soubor. Vyplývá z toho také to, že u stavového serveru bude větší problém se zotavením dat po pádu serveru.

Celý problém se znásobí, pokud budeme chtít implementovat vyrovnávací paměť klienta. Ta je sice výhodná z důvodu zrychlení při časté práci se stejnými daty, čímž snižuje zátěž serveru a sítě. Ale vyplývá z toho problém konkurenčního zápisu a čtení stejných dat více klienty. Každý totiž upravuje vlastní kopii v paměti a zápis bloků dat jednotlivých klientů způsobí nekonzistenci souboru. Existuje několik možností řešení. Například používat relační zápis, kde se změny provádějí u klienta a na server se promítnou až po uzavření souboru. Sice se stále přepíší změny ostatních, ale celkově data budou konzistentní. Můžeme také zakázat úpravy, povolíme pouze čtení a vytvoření souboru. Přetrvá problém při vytváření stejného souboru více uživateli, ale data budou konzistentní. Dalšími metodami je použití transakcí s atomickými úpravami a zamykáním souborů nebo použití centrálního řídicího serveru. Ten bude kontrolovat požadavky jednotlivých klientů a rozhodovat jak naloží s požadavky na stejný soubor. To může být ale úzké hrdlo celého systému.

1.1.2 Architektura

Většina systémů využívá, jak už bylo řečeno, architekturu distribuce a rozdělení dat na více fyzických serverů (uzlů). Z pohledu zvýšení ochrany dat a jejich dostupnosti je výhodné mít data duplikována. V případě velmi důležitých dat s potřebou jejich vysoké dostupnosti jsou distribuované systémy duplikující data po síti velmi výhodné řešení. Systémy jsou většinou tolerantní k chybám a v případě výpadku datového uzlu směrují požadavky jinému uzlu.

Duplikace dat nemá výhodu pouze ve zvýšení dostupnosti a bezpečnosti, ale také bývá využívána pro škálování jejich diskového výkonu a propustnosti sítě. V případě vysokého počtu klientů a potřeby číst nebo zapisovat velké množství dat najednou (například obrovské výpočetní clustery) jsou nasazovány stovky až desetitisíce uzlů. Ty obsahují distribuovaný souborový systém, který se chová jako jeden celek, a požadovaný diskový výkon se rozprostře mezi ně. Soubory se mohou na další uzly buďto duplikovat, nebo se rozdělí na fragmenty, ty se pak ukládají na různé servery. Při čtení pak není zatěžován jeden server a jeden uzel sítě, ale zátěž se rozprostře.

Pro duplikování dat se využívá několik metod. Duplikace na vyžádání, u které v případě požadavku systém duplikuje daná data, většinou na jiné uzly. Takzvaná líná duplikace, kde data mají většinou nastavený atribut s počtem, kolikrát mají být duplikována. Server poté v čase nízké zátěže duplikuje data na další uzly tak, aby neomezoval probíhající akce. Skupinové duplikace, kdy jsou zápisy rozesílány najednou (nebo postupně) na více uzlů. Při duplikacích může dojít k problémům se synchronizací. Klientovi se může stát například to, že provede změny na souboru a poté bude chtít opět číst tento soubor. Ale získá data neupravená, protože požadavek odejde na uzel, který ještě změněná data neobdržel. Toto se musí řešit složitějšími algoritmy. Některé systémy umí navíc vytvářet duplikace dat pouze pro čtení.

Pro zvýšení výkonu má také většina systémů oddělena metadata od dat na vlastním serveru (serverech). Požadavky na data budou procházet nejdříve takzvaným metadata serverem. To v případě velkého počtu menších souborů, kde nejvíce výkonu zabere čtení právě metadat, ušetří zátěž datovým serverům. [1]

1.2 Přehled systémů

Distribuovaných síťových souborových systémů existuje široká škála, většina z nich byla buďto vyvíjena jako universitní projekt, nebo se jedná o projekty velkých korporací pro jejich vlastní užití. Některé z nich potom takovýto systém vydali jako volně šiřitelný pod nějakou veřejnou licencí. Ze seznamu známých souborových systémů bylo pro testování vybráno následujících sedm, které jsou známé, hojně využívané a dostupné většinou pod licencí GPL. [2]

1.2.1 NFS

Network File System je protokol původně vyvinutý firmou Sun Microsystems (1984), která ho později vydala jako veřejný dokument RFC [3]. Je založen, jako mnoho dalších, na systému vzdálených volání ONC RPC (Open Network Computing Remote Procedure Call).

Z počátku byl vyvíjen jako co nejjednodušší bezestavový systém typu server-klient umožňující přístup k souborům po síti protokolem UDP. Nedisponoval zamykáním souborů a postrádal bezpečnost. Zamykání souborů bylo umožněno externím „zamykacím“

serverem. Verze 3 přidala podporu síťového protokolu TCP, což umožnilo využívat NFS v sítích WAN, kde může docházet k větším výpadkům síťových paketů.

NFSv4 výrazněji vylepšuje výkon, přidává možnosti zabezpečení pomocí Kerbera/LDAP, dokonce i stavový protokol. Oproti předchozím verzím exportuje (sdílí) okolním klientům jen jeden kořenový adresář.

Nejnovější (leden 2010) verze protokolu 4.1 přináší paralelní rozšíření (pNFS), které umožňuje využít tento systém v clusterových prostředích, kde je v rámci škálování nasazeno více serverů NFS, které se chovají jako jeden.

1.2.2 SMB

Protokol SMB, známý také jako CIFS, je pravděpodobně nejvyužívanějším síťovým souborovým systémem, a to díky rozšířenosti v operačních systémech Microsoft Windows. Byl původně vyvinut firmou IBM pro operační systém DOS. Firma Microsoft ho do všech dalších verzí svého operačního systému reimplementovala s vlastními úpravami a novými vlastnostmi.

Existuje také projekt Samba vydaný pod licencí GNU GPL. Ten byl vytvořen za pomoci veřejně vydaných specifikací SMB/CIFS, ale také reverzního inženýrství. Přináší ostatním operačním systémům kompatibilitu se servery a klienty systémů Microsoft Windows.

Samba implementuje desítky služeb a protokolů, takže mimo klasického sdílení souborů dokáže sdílet tiskárny a může se dokonce chovat jako klient/server doménového systému Microsoft NT nebo Active Directory. [4]

1.2.3 AFS

AFS (Andrew File systém) byl vyvinut universitou Carnegie Mellon University ve spolupráci s firmou Transarc (později IBM). Jednalo se o součást kompletního distribuovaného řešení Andrew Project, který obsahoval tento souborový systém, vlastní okenní manažer a několik kancelářských aplikací (textový a tabulkový procesor, emailový klient apod.).

Firma IBM pak vydala tento komerční souborový systém jako open source pod názvem OpenAFS. V dnešní době existují čtyři implementace tohoto systému: původní Transarc

(IBM), OpenAFS, Arla a také implementace přímo v linuxovém jádře od verze 2.6.10, začleněná firmou Red Hat.

Aktuálně nejvíce vyvíjeným a nejkompletnějším volně dostupným řešením je projekt OpenAFS. Ten je dostupný pro velké množství operačních systémů včetně Unixu, Linuxu, Mac OS X, ale také Microsoft Windows, dokonce i pro nejnovější verze Seven a Server 2008.

Jelikož se tento systém většinou využívá v rozsáhlých prostředích (university, výzkumné ústavy, veliké organizace), umožňuje logicky organizovat klienty do domén, nazývaných buňky. Z oblasti bezpečnosti lze zmínit, že pro autentizaci využívá systému Kerberos a také že implementuje ACL pro uživatele a skupiny. Pro zvýšení výkonu a škálovatelnosti klient obsahuje vyrovnávací paměť.

Architektura systému je typu server-klient, přičemž serverů může být samozřejmě více. Serverem není jediný proces, ale jedná se o skupinu specializovaných procesů. To umožňuje na některých serverech provozovat pouze potřebné služby (například pouze zálohovací server). Možnými serverovými procesy jsou:

- Basic OverSeer Server (BOS Server) hlídající ostatní procesy, zda běží v pořádku.
- File Server zajišťující manipulaci s požadovanými daty.
- Protection server obstarávající nastavování a dodržování práv ACL.
- Volume Server zařizující veškerou manipulaci se svazky, například přesun svazku na jiný stroj z důvodu kapacity nebo zátěže, což se provádí nad běžícím systémem bez ovlivnění klientů.
- Volume Location Server spravující seznam umístění svazků na jednotlivých souborových serverech, což je důležité pro transparentnost umístění z pohledu klientů, jelikož se svazky mohou přesouvat či replikovat.
- Update Server je volitelný proces starající se o to, aby na každém serveru běžela stejná verze jednotlivých procesů, což je důležité pro bezchybnou kompatibilitu; v případě nesouhlasných verzí update server doručí a zajistí nainstalování správné verze.
- Backup Server umožnující administrátorovi zálohovat nebo obnovovat definované svazky (nebo sety svazků) na zálohovací zařízení, většinou na vysokokapacitní

magnetické pásky. Jsou k dispozici přírůstkové zálohy, tudíž je možné vrátit svazky při pravidelných zálohách do stavu určitého data, aniž by tato zálohovaná data zabírala několikanásobnou velikost. Tomu také napomáhá možnost nastavení expirace některých záloh.

- Salvager je od předešlých procesů odlišný. Spouští ho BOS server pouze v případě chyby (pádu) File serveru, nebo Volume Serveru. Salvager se poté snaží opravit chyby v datech, které mohly nastat.

Jak z předešlého textu vyplývá, v jednotlivých buňkách může být více svazků na jednotlivých diskových oddílech jednoho serveru, nebo mohou být rozprostřeny na více serverech. Nejedná se o logické diskové svazky, ale o hlavní organizační jednotku tohoto systému. Takovým základním organizačním rozdělením by mohlo být například vytvoření svazku pro každý domovský adresář jednotlivých uživatelů. Dává to administrátorovi možnost efektivně přesouvat jednotlivé svazky mezi disky nebo servery. To může být potřeba například z důvodu docházející kapacity, nebo vyrovnání zátěže, dle požadavků jednotlivých uživatelů. [5], [6]

1.2.4 Lustre

Lustre je distribuovaný souborový systém, vyvíjený firmou Sun Microsystems (nyní Oracle), určený pro obrovské clusterové systémy. Dokáže obsluhovat desítky tisíc klientů, kterým může nabízet kapacitu dat v řádu petabajtů, rozložených až na 10 000 datových serverech. Díky vysoké škálovatelnosti má pak celý systém propustnost dat v řádech gigabajtů za sekundu, proto je využíván na velkém počtu nejvýkonnějších clusterových systémů na světě.

Systém byl vydán pod open source licencí GNU GPL a byl adaptován několika firmami dodávajícími kompletní linuxová enterprise řešení, například Red Hat Enterprise Linux, SUSE Linux Enterprise Server nebo vlastní Oracle Enterprise Linux.

Je založen na architektuře tří prvků. Prvními jsou servery Object storage (OSS), což jsou servery ukládající vlastní data, a může jich být 1–10 000. Nedílnou součástí jsou Metadata servery (MDS), které uchovávají metadata k daným datům. V aktuální stabilní verzi mohou být dva (hlavní a záložní), ale v připravované verzi se počítá s rozšířením až na 100. Poslední prvek tvoří samozřejmě klienti, kterých mohou být až desítky tisíc.

Systém obsahuje mnoho mechanismů pro zachování vysoké dostupnosti dat, v případě nutnosti restartovat některý server nebo v případě výpadku, takže je systém schopen dále běžet a obsluhovat klienty. Dále oproti konkurencím nativně podporuje a umí efektivně využít několik druhů síťových technologií, jako je například Infiniband, TCP/IP po Ethernetu, Myrinetu, technologii Quadrics a další. Umí využít, pokud je dostupná, také technologii RDMA, což je vzdálený přímý přístup do paměti, který snižuje zátěž na procesor a zvyšuje propustnost sítě. [7], [8]

1.2.5 MooseFS

Je systémem vyvíjeným firmou Gemius SA původně jako uzavřený kód a v roce 2008 vydaný jako otevřený pod licencí GPL. Od začátku se snaží být POSIX kompatibilním, škálovatelným systémem tolerantním k chybám.

Pro diskové operace se tedy chová jako běžný unixový souborový systém, obsahuje adresářový strom, atributy, má možnost vytvářet speciální soubory (roury, odkazy, zařízení) atd.

Oproti ostatním systémům však disponuje několika dalšími funkcemi. Jednou z nich je odpadkový koš, který uchovává smazané soubory po určenou dobu. Dále umožňuje jednoduše nastavovat počty kopií dat, ve kterých se mají replikovat na další servery. Obsahuje také informační webové rozhraní, ve kterém se správce jednoduše dozví podrobný stav jednotlivých serverů, jejich oddílů, počtů replikací jednotlivých bloků dat, operací a také klientů. Rozhraní také obsahuje generované grafy zátěže procesoru a jednotlivých druhů operací nad daty. Grafy jsou dostupné jak pro hlavní server, tak pro data servery. Lze přepínat časový rozsah grafů, od hodin až po měsíce, tudíž je k dispozici ucelený přehled využívání systému.

Architektura je zde obdobná jako u předešlých dvou systémů, pouze s drobnými změnami a názvy jednotlivých částí:

- Master server je jeden server řídící celý souborový systém a uchovávající veškerá metadata pro všechny soubory.
- Chunk servery, kterých může být více (minimálně jeden), skladují holá data rozdělená na určité bloky. Pokud má být soubor replikovaný, tak jej také mezi sebou v daném počtu synchronizují.

- Záložní metadata servery si periodicky stahují z hlavního serveru soubor s metadaty a také si ukládají postupné informace o změnách, aby byly schopné v případě výpadku zaujmout pozici master serveru. To je v tomto systému nová funkce, která se musí prozatím konfigurovat za použití protokolu CARP a pomocných skriptů.
- Klientské počítače s běžícím procesem mfsmount, který zařizuje veškerou komunikaci s master serverem a chunk servery, kterým zasílá fragmenty dat.

Aplikace mfsmount je postavena na mechanismu FUSE, který umožňuje implementovat souborový systém v uživatelském prostoru. Není tedy třeba jaderných ovladačů, ani úprav jádra. Klientem může tudíž být jakýkoli operační systém, který má implementován FUSE (GNU/Linux, FreeBSD, MacOS X, …). [9]

1.2.6 CODA

CODA je systémem vyvíjeným od roku 1987. Jedná se o výzkumný projekt, který je vyvíjen na stejné universitě jako AFS a dokonce z tohoto systému vychází, proto je koncept velice podobný. Vývoj probíhá dodnes a snaží se vytvořit robustní distribuovaný systém pro komerční použití.

Oproti ostatním systémům se snaží zaměřit i na mobilní uživatele. Proto disponuje možností pracovat s daty i po odpojení sítě. Data se pak po opětovném připojení sesynchronizují, přičemž může častěji docházet ke konfliktům, které musí systém řešit. Další vlastností je přizpůsobivost rychlosti sítě pro klienty připojené pomalejší linkou, což je v sítích WAN běžné.

Zatímco AFS podporuje od každého oddílu vždy pouze jednu kopii s možností zápisu a ostatní klonované kopie slouží pouze pro čtení, CODA umožňuje vytvářet více kopií, do kterých se synchronizují také zápisy. Při tomto může také docházet ke konfliktům, kdy například dva uživatelé upravují stejná data, ale od každého z nich odejde požadavek na jiný server. Servery pak mezi sebou detekují konflikt, který se následně snaží vyřešit. V případě, že problém není automaticky vyřešitelný, jsou data označena atributem a konflikt musí vyřešit uživatel s dostupnými oprávněními.

Stejně jako u AFS, jsou i zde klienti a servery organizováni do buněk. Buňka se vždy skládá z minimálně jednoho serveru (nazývaný Vice), mohou jich však být i stovky. Vždy jeden z nich je určen jako server SCM, který je určen pro veškeré nastavování a úpravy,

které se pak promítají na ostatní servery. Oproti systémům NFS nebo Samba nesdílí CODA přímo část adresářové struktury lokálního filesystému. Jelikož potřebuje daleko více metadat z důvodu replikací, operací při odpojení a také komplexní obnovy dat, ukládá soubory (pojmenované číselnými hodnotami) v adresářích standardně /vicepX (kde X se nahrazuje písmeny a až z) a metadata ukládá zvlášť do RVM.

RVM znamená obnovitelná virtuální paměť. Je to knihovna zprostředkovávající mechanismus pro transakční provádění atomických změn na disku za pomocí mapování vyhrazené paměti na disku (soubor nebo lépe diskový oddíl RAW) do fyzické operační paměti. Pomocí této knihovny jsou ukládány metadata jak na serveru, tak u klienta, který obsahuje navíc ještě klientskou aplikaci s vlastním manažerem vyrovnávací paměti, dohromady nazývané jako Venus. Klient se servery komunikuje, jako většina těchto systémů, pomocí vzdálených volání procedur RPC. [10]

1.2.7 Tahoe-LAFS

Tahoe-LAFS je dalším z dostupných distribuovaných systémů, vyvinutý a vydaný jako volně dostupný systém firmou Allmydata. Je zaměřen na dlouhodobé a bezpečné uložení dat dostupných po síti.

Na bezpečnost a ochranu dat je zde kladen velký důraz. Data jsou proto fyzicky na disku uložena v zašifrované podobě, takže ani poskytovatel, nebo případně administrátor, nemá možnost data ani číst, natož měnit. Přenos dat probíhá samozřejmě také šifrovaně. Veškerá data mají určenu hodnotu *N*, na kolika různých serverech mají být replikována (standardně 10) a hodnotu minimálního počtu replikací *k*, která je nezbytná pro správné získání původních dat (standardně 3). Může tedy dojít k výpadku *N–k* serverů, aniž by došlo k omezení dostupnosti daných dat. Pokud je z jakéhokoli důvodu méně dostupných serverů, než je požadovaných replikací, uloží se na některý ze serverů více kopií stejných dat.

Architektura tohoto systému je typu klient-klient. Obsahuje takzvané Key-value servery (které ukládají zašifrované části dat) a jeden tzv. Introducer. To je server, který má informace o všech uzlech, a k němu se klient na začátku připojí, aby získal adresy uzlů. Ke každému z nich se pak připojí. Introducer je tedy jediným slabým článkem, protože není nijak

duplikován, což ale v tomto případě není až takový problém. Klienti se ho dotazují pouze při připojení, pak již komunikují přímo s key-value servery.

Do systému se dají uložit dva typy souborů, zde nazývaných immutable a mutable. První z nich mají takovou vlastnost, že jakmile je takový soubor nahrán na servery, není možné ho modifikovat. Oproti tomu mutable soubory je možné jak číst tak upravovat. [11]

2 Metodika testování

2.1 Konfigurace testovacích PC

Důležitým prvkem je zde síťový segment, na kterém je výkonnost těchto systémů velice závislá. Na výběr byla nejběžnější technologie stomegabitového Fast Ethernetu s kabeláží typu UTP Cat.5E a centrálním přepínačem ES-3108P firmy Edimax 10/100Mbit, 128 KiB vyrovnávací paměť. Pro testování síťových souborových systémů by bylo zajímavé srovnání při nasazení gigabitového Ethernetu. Bohužel ale sestava číslo 2 neobsahuje gigabitovou síťovou kartu a přepínač tuto technologii také neumožňuje. Proto byla pro jednotné testování zvolena technologie 100Mbit Fast Ethernet.

Testováno bylo na třech počítačových sestavách výkonově dostatečně rozdílných. První je starší značkový notebook s pomalejším jednojádrovým procesorem. Druhá je sestava s procesorem AMD Opteron určeným pro servery. Třetí sestavou je osobní počítač s dvoujádrovým procesorem Intel a větší kapacitou fyzické paměti.

Předpokladem ale je, že tyto parametry nebudou ani u nejslabšího počítače omezující. Omezujícím prvkem bude pravděpodobně síťový segment.

2.1.1 Hardware

- **Sestava 1:**
  - notebook IBM ThinkPad X31,
  - procesor – Intel Pentium M, 1 600 MHz, 1 MiB vyrovnávací paměti,
  - RAM – 2× 512 MiB, non-ECC DDR SDRAM,
  - síťová karta – Intel Corporation 82801DB PRO/100 VE,
  - HDD – 2,5 ″, 100 GB, 5 400 rpm.
- **Sestava 2:**
  - základní deska – MSI K8N Neo 2 Platinum, nVidia nForce 3 Ultra, patice 939,
  - procesor – AMD Opteron 144, jedno jádro na frekvenci 2 600 MHz, 1 MiB vyrovnávací paměti,

- RAM – 2× 512 MiB, non-ECC DDR SDRAM, 400 MHz, dvoukanálové,
- LAN 1 – 10/100/1000 Fast Ethernet by Marvell 88E1111 PHY,
- LAN 2 – 10/100/1000 Fast Ethernet by Realtek 8110S (použita pro testování),
- HDD 1 – 3,5 ″, 200 GB SATA Maxtor 6L200M0, 7 200 rpm,
- HDD 2 – 3,5 ″, 320 GB PATA Western Digital WD3200AAKB, 7 200 rpm.

- **Sestava 3:**
  - základní deska – MSI G33/P35 Neo, Intel Bearlake, patice 755,
  - procesor – Intel Core2Duo E7200 (2 500 MHz), 2 jádra, 3 MiB L2 vyrovnávací paměti,
  - RAM – 2× 2 GiB,
  - síťová karta – Realtek RTL8168B Family PCI-E Gigabit Ethernet NIC,
  - HDD – 3,5 ″, 640 GB Western Digital WDC WD6400AAKS, 7 200 rpm, SATA II.

2.1.2 Software

- **Server**
  - Jako server byla vždy použita distribuce Ubuntu Server 9.10 s distribučním jádrem Linux 2.6.31-14-generic (SMP, i686).
  - Jednalo se o čistou instalaci doplněnou o balíky serveru SSH pro snadnější přístup.
  - Pro data byl na každém PC vytvořen vlastní diskový oddíl o velikosti 10 GiB naformátovaný na aktuálně nejběžnější linuxový souborový systém ext3, oddíl byl připojen pomocí standardních parametrů.
- **Klient**
  - Klientem byl vždy jeden ze zbylých PC a obsahoval distribuci Ubuntu Desktop 9.10 s jádrem Linux 2.6.31-21-generic (SMP, i686).
  - Zde se nejednalo o čistou instalaci, ale o běžně užívaný systém, tudíž na něm zároveň běžely některé nadbytečné služby, které ale testování ovlivnily v nejmenší možné míře.

2.2 Testovací nástroje

Pro testování souborových systémů lze nalézt velké množství dostupných nástrojů, které jsou si nakonec velice podobné, protože používají obdobné metody testování. Byla vybrána aplikace Bonnie++ z důvodu přehledných a jednoduše porovnatelných výstupů.

Pro testování souborových systémů je doporučováno testovat na datech o velikosti minimálně dvojnásobku kapacity operační paměti, čímž se minimalizuje zkreslení způsobené využitím vyrovnávací paměti.

2.2.1 Bonnie++

Bonnie++ je do jazyka C++ přepsaná a vylepšená verze aplikace Bonnie určená pro testování unixových souborových systémů. Testuje tyto operace: rychlost čtení a zápisu, počet vyhledávání za sekundu a počet operací nad metadaty za sekundu.

Velikost testovaného souboru, pokud nezadáme pevně jinou, určí Bonnie++ sám jako dvojnásobek operační paměti. To je v případě dvou testovacích sestav 1 GiB a ve třetím případě 4 GiB. Ale jelikož je použito 32bitové jádro bez zapnutého režimu PAE, je pro systém použitelných zhruba 3,3 GiB operační paměti.

Testování bylo prováděno pouze se zadáním přepínače -q, který zapříčiní výpis všeho, kromě posledního řádku na chybový výstup. Řádek obsahuje výstupní hodnoty oddělené čárkami, takže jej můžeme ze standardního výstupu přesměrovat do souboru. Ten pak použijeme k vygenerování tabulky do souboru txt nebo html pomocí skriptů bon_csv2html a bon_csv2txt. V testech se můžeme místo nějaké hodnoty setkat se značkou „+++++“, což znamená, že daný test proběhl příliš rychle. V případě testování jednoho systému bychom pomocí parametrů upravili průběh testování tak, abychom získali i tyto hodnoty, ale pro srovnání s ostatními systémy je nutné dodržet stejné parametry testování. [12]

Použitý příkaz:

```
bonnie++ -q > vystup.csv
```

3 Implementace a testy

3.1 Všeobecně

Jak na straně serveru, tak klienta byl použit operační systém Ubuntu, který je odvozen od systému Debian, používá tedy balíčkovací systém typu deb. Z pohledu co největší jednoduchosti byla snaha instalovat DFS vždy nejdříve ze standardních distribučních repositářů nebo repositáře vývojářů daného systému. Pro tuto instalaci je možné použít místo uváděné aplikace aptitude jakýchkoli nástrojů systému Ubuntu (dpkg, apt-get, aptitude, Synaptic a další). Pokud se v těchto zdrojích nenachází, bývá k dispozici buďto binární instalátor nebo častěji zdrojové kódy, které si musíme pro náš systém zkompilovat. Vysvětlení postupu kompilace není předmětem této práce. Více viz [18].

Většina systémů využívá pro adresaci ostatních počítačů místo adres IP doménová jména počítačů, proto je potřeba mít v síti zprovozněn překlad jmen na adresy. Pro malé sítě je to možné realizovat pomocí souborů hosts na každém z PC obsahujících adresy a jména. To je ale zastaralá, při přidávání nových strojů náročná a pro větší sítě značně nevýhodná metoda, proto se toto řeší tzv. servery DNS. V testované síti byl přítomen linuxový směrovač obsahující aplikaci dnsmasq, který si uchovává jména a adresy, které přiřadil jeho server DHCP a obstarává jejich překlad pro přicházející dotazy.

V případě některých serverů DFS mohl nastat problém se souborem hosts, který lokální jméno přeložil na adresu 127.0.0.1, se kterou neuměly DFS z nějakého důvodu pracovat. Proto bylo lokální jméno z tohoto souboru odstraněno a dotazy na toto jméno se tudíž po nenalezení směrovaly na server DNS.

Příkazy příkazové řádky a některé konfigurační volby jsou pro přehlednost uváděny na šedém pozadí, nebo pokud jsou přímo v textu, jsou uvedeny v uvozovkách. Příkazy jsou uváděny spolu s prefixem shellu (např. server#, což znamená, že je příkaz prováděn na serveru, a # značí, že je prováděn pod uživatelem root). V systému Ubuntu z důvodu bezpečnosti není tomuto uživateli povoleno přihlásit se standardní cestou, proto pro provedení příkazu pod tímto uživatelem před ním použijeme příkaz sudo nebo provedeme příkaz „sudo -s“, případně „sudo su“, který nás na tohoto uživatele přihlásí.

3.2 NFS

Implementací tohoto protokolu existuje více. V tomto případě se bude instalovat balík z distribučního repositáře s názvem nfs-kernel-server, jedná se o aplikaci běžící v režimu jádra a implementující protokol NFS do verze 4. Stabilní verze protokolu 4.1 s možností paralelizace aktuálně k dispozici není.

3.2.1 Server

Instalace balíku a jeho závislostí a vytvoření adresáře který budeme sdílet:

```
server# aptitude install nfs-kernel-server
server# mkdir /share
```

Úprava hlavního konfiguračního souboru /etc/exports. Zde jsou po řádcích určeny exportované adresáře s klienty (IP adresa/rozsah nebo doménové jméno), kteří si je mohou připojit, a vlastnostmi exportování. Export může vypadat například takto:

```
/share     klient(rw,async,fsid=0,insecure)
```

Exportujeme pouze jednomu počítači (jménem „klient“) s možností zápisu, asynchronním režimem (server potvrdí úspěch už před samotným zápisem, což zvýší výkon, ale také riziko problému, pokud zápis selže). Volba fsid označuje kořenový sdílený adresář a insecure znamená, že nebudeme využívat žádné ověřování (Kerberos). Po úpravě konfiguračního souboru restartujeme server, např.:

```
server# service nfs-kernel-server restart
```

3.2.2 Klient

Nainstalujeme balík klienta a vytvoříme adresář pro připojení:

```
klient# aptitude install nfs-common
klient# mkdir /nfs
```

Upravíme volby konfiguračního souboru /etc/default/nfs-common tak, aby se spouštěl démon idmapd, který překládá vlastníky a skupiny připojených systémů, a vypneme GSSD, který je třeba pro ověřování Kerberem:

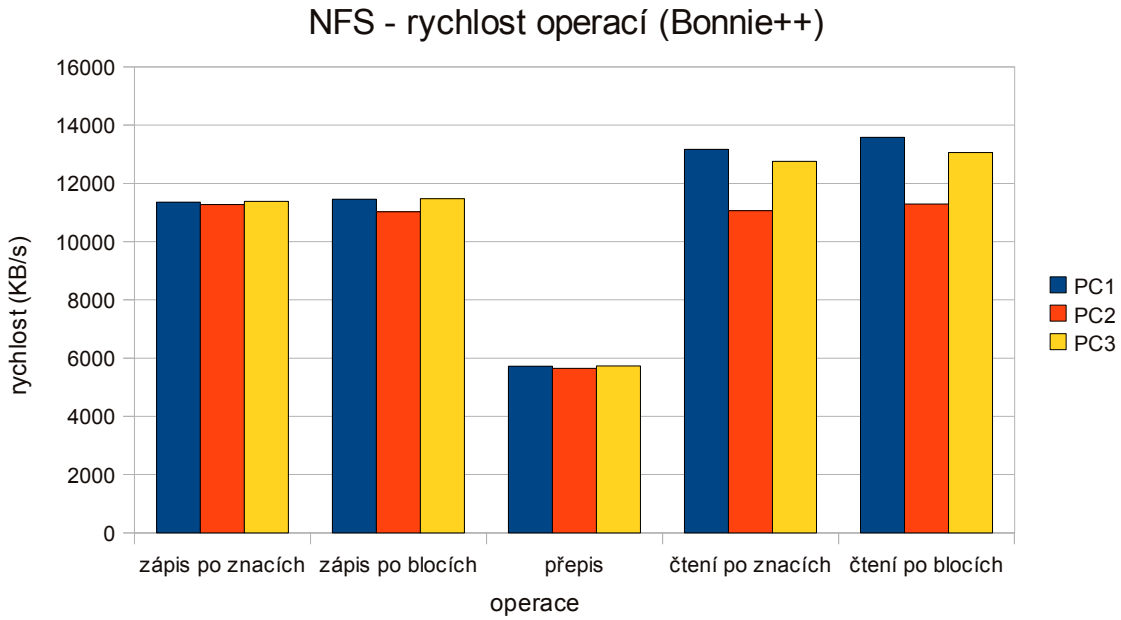
```
NEED_IDMAPD=yes
NEED_GSSD=no
```

A nakonec připojíme vzdálený adresář (dlouhodobě je vhodné přidat volbu do /etc/fstab spolu s volbou „auto“, aby se adresář připojoval automaticky během startu):

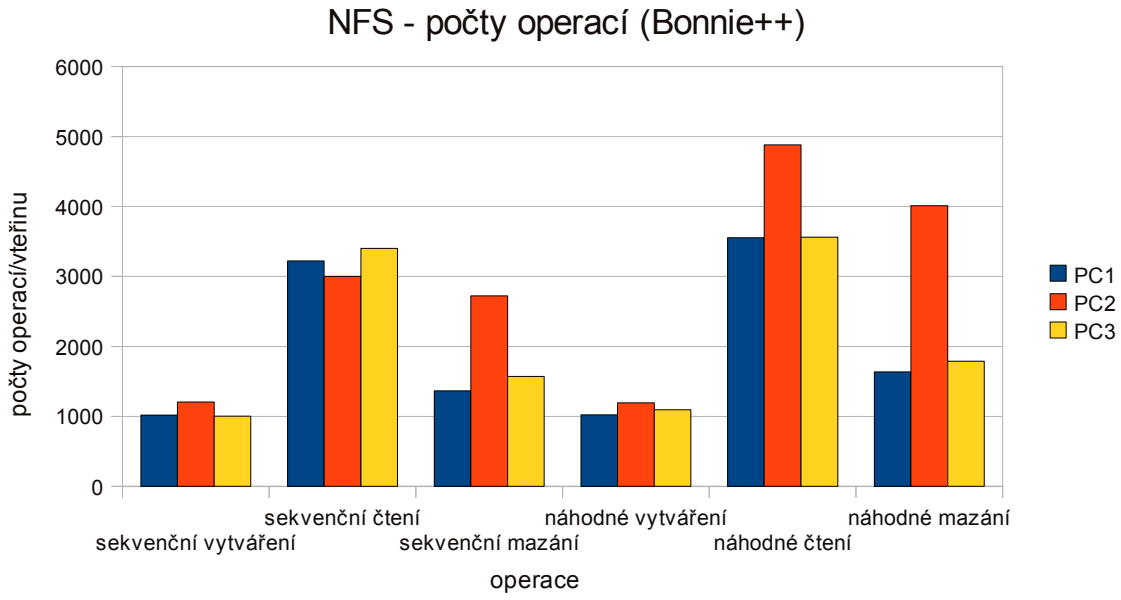
```
klient# mount -t nfs4 -o rw server:/ /nfs
```

[13]

3.2.3 Benchmarky



Ilustrace 1: NFS – rychlost operací



Ilustrace 2: NFS – počty operací

3.2.4 Hodnocení

Časem prověřený systém, jednoduchý na nainstalování, konfiguraci i požití a je stále vyvíjen a vylepšován. Nevyžaduje žádnou dodatečnou správu.

Dle výsledných hodnot a sestavených grafů lze říci, že je systém výborně implementován a vyladěn, protože přenosové rychlosti dosahují téměř maxima stomegabitové sítě, což je zhruba 12 500 kilobajtů za sekundu, a to na všech sestavách. Při čtení je tento limit navíc nepatrně překročen díky vyrovnávací paměti u klienta. Druhý graf zdánlivě vykazuje vyšší výkon druhého počítače. To je ale v tomto případě způsobeno větší vyrovnávací pamětí počítače číslo 3, ze kterého bylo testováno. Pro vyšší objektivnost testů by bylo zapotřebí nezávislého počítače využívaného jako klienta pro všechny testy.

3.3 SMB

Jak již bylo řečeno, veřejně dostupnou implementací tohoto protokolu je projekt Samba, který má kromě sdílení souborů spoustu dalších funkcí, kterými se ale zabývat nebudeme.

3.3.1 Server

Distribuční repositář obsahuje balíky jak serveru, tak klienta, tudíž není třeba žádných úprav a stačí nainstalovat:

```
server# aptitude install samba
```

Hlavním konfiguračním souborem je /etc/samba/smb.conf, kde na konec přidáme nastavení pro exportovaný adresář, tento konfigurační soubor je velmi rozsáhlý a nabízí spoustu nastavení, nám ale postačí toto jednoduché nastavení:

```
[Public]
path = /share
public = yes
writable = yes
```

Public bude názvem sdíleného adresáře v síti, path je cesta k lokálnímu sdílenému adresáři. Druhý výskyt „public“ znamená, že bude adresář viditelný při prohlížení sdílených adresářů a writable umožní zápis.

Standardním režimem samby je režim user, tudíž je pro připojení nutná autentizace a je tedy vyžadován účet jak v sambě tak i v systému (/etc/passwd).

Nyní přidáme uživatele s jeho heslem a server restartujeme.

```
server# smbpasswd -a smbuser
server# service samba restart
```

3.3.2 Klient

Nainstalujeme balík klienta spolu s balíkem obsahujícím dodatečné klientské aplikace.

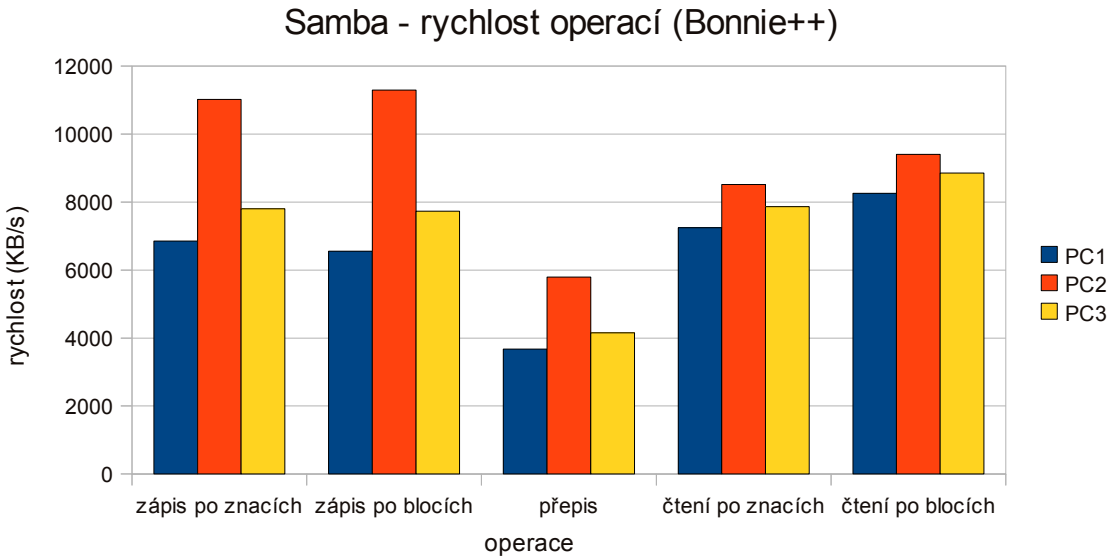
```
klient# aptitude install samba-common smbfs
```

Nyní už můžeme vytvořit adresář a připojit do něj vzdálený systém:

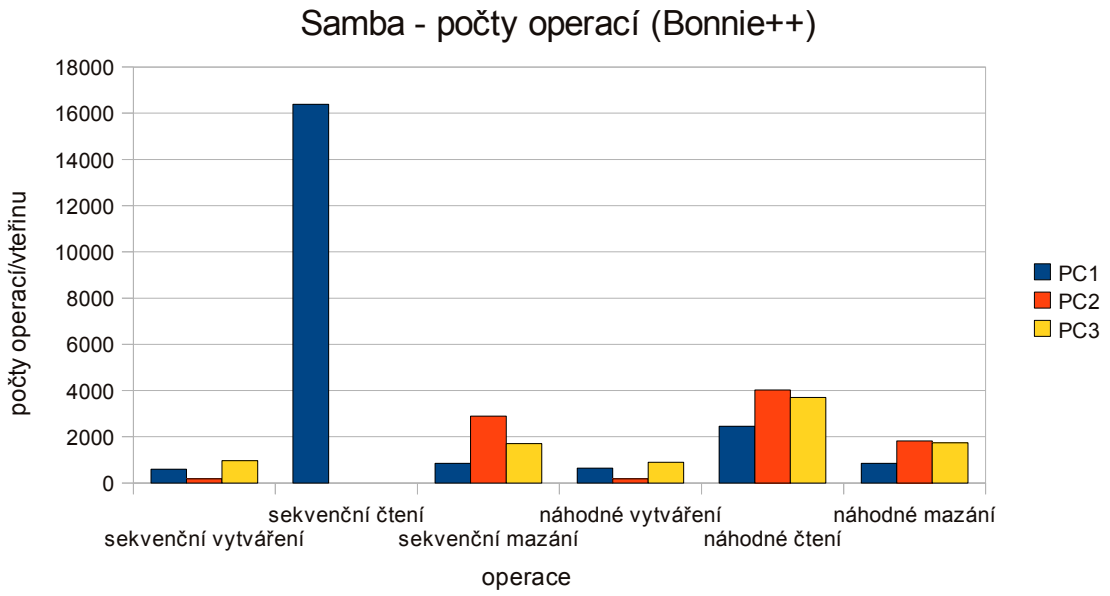
```
klient# mkdir /mnt/samba
klient$ mount -t cifs //server/Public /mnt/samba -o user=smbuser
```

Musíme počítat s tím, že bez dodatečného nastavení mapování je vyžadován jak na straně serveru (v systému a v smbpasswd), tak na straně klienta v systému uživatel se stejným jménem, jinak nebudou správně fungovat oprávnění. Standardně se pak v případě nenalezení takového uživatele mapují veškeré operace na uživatele nobody a skupinu nogroup, které mají dosti nízká oprávnění. [14]

3.3.3 Benchmarky



Ilustrace 3: Samba – rychlost operací



Ilustrace 4: Samba – počty operací

3.3.4 Hodnocení

Tento systém je opět velice jednoduché nainstalovat a nakonfigurovat pro základní použití, které navíc nebude potřebovat žádnou průběžnou správu administrátorem. Navíc nabízí popsané možnosti adresářové služby a další.

Výsledky testu Bonnie++ vykazují poměrně nižší hodnoty než u systému NFS, se kterým bývá často srovnáván. V druhém grafu (Ilustrace 4) je v sekvenčním čtení uveden pouze jeden sloupec pro nejpomalejší sestavu, na zbylých sestavách proběhl test příliš rychle a Bonnie++ ho nezměřilo. Tak vysoké hodnoty jsou pravděpodobně důsledkem využití vyrovnávací paměti, bohužel pouze v jednom typu testů.

3.4 AFS

Pro testování byla vybrána, jak již bylo řečeno v úvodu o tomto protokolu, nejkompletnější, funkčně nejstabilnější a nejvyužívanější implementace OpenAFS. Jak klientské, tak serverové balíky se nacházejí, byť v nižší verzi, ve standardních repositářích distribuce Ubuntu. Pokud bychom chtěli testovat nejnovější verze, jsou k dispozici samozřejmě zdrojové kódy.

Jak klienti, tak všechny servery vyžadují pro svůj běh jaderný modul. Ten se většinou musí ručně kompilovat spolu se zdrojovými kódy daného jádra. Vývojáři OpenAFS ale vy-

užili možnosti dynamického nahrávání jaderných modulů (DKMS). Je tedy potřeba mít nainstalovaný balíček dkms, hlavičkové soubory aktuálního jádra (linux-headers-<verze-jádra>), standardní aplikace a knihovny nutné pro kompilaci zdrojových kódů (gcc, libc, make, …). Aplikace DKMS pak při instalaci OpenAFS a při každé aktualizaci jádra automaticky zkompiluje daný modul oproti hlavičkovým souborům a modul nainstaluje, takže se o nic nemusíme starat.

3.4.1 Server

Systém potřebuje pro svou činnost dvě dodatečné služby. První je autentizační služba Kerberos, kterou se budou mezi sebou autentizovat jak klienti, tak servery. A druhou je časová synchronizační služba nutná jak pro samotny souborový systém, tak pro službu Kerberos. Časový rozdíl systémů větší než několik sekund, by mohl způsobit výkonový propad některých operací, případně by mohlo dojít k odmítnutí autentizace.

Pro službu autentizace byla použita implementace Kerbera verze 5 s názvem krb5 (balíky krb5-kdc, krb5-admin-server) a pro časovou synchronizaci byl na serveru nasazen démon ntp (balík ntp-server), který synchronizoval čas z několika serverů na internetu a aktuální synchronizovaný čas vysílal do podsítě pomocí všesměrového vysílání. Nastavení těchto služeb ale není předmětem této práce, proto nebude podrobněji rozebíráno.

Instalaci samotného serveru a jeho závislostí provedeme příkazem:

```
server# aptitude install openafs-fileserver openafs-dbserver \
  openafs-client libpam-afs-session
```

Instalátor jednotlivých balíků se postupně ptá na konfigurační informace: buňku (většinou doménové jméno), velikost vyrovnávací paměti, jména databázových serverů a zda se má klient spustit nyní a po každém startu systému (při instalaci prvního serveru je vhodné toho nenastavit, protože server ještě není nakonfigurován).

Další postup již předpokládá nastavenou službu Kerberos spolu s nastavením ověřování pomocí modulu PAM. Důležité je vytvoření minimálně dvou identit v Kerberu, administrátora a identity pro komunikaci mezi serverovými procesy. Pro druhou identitu vyexportujeme vytvořený klíč který budeme dále využívat.

Prvně spustíme řídící server bez nutnosti autorizace a rovnou nastavíme základní údaje:

```
server# bosserver -noauth
server# bos setcellname server.localdomain localdomain -noauth
```

Nastavení databázového serveru – vytvoření backup, protection a volume location modulů:

```
server# bos create server.localdomain buserver simple \
  /usr/afs/bin/buserver -noauth

server# bos create server.localdomain vlserver simple \
  /usr/afs/bin/vlserver -noauth

server# bos create server.localdomain ptserver simple \
  /usr/afs/bin/ptserver -noauth
```

Přidáme nastavení pro souborový server:

```
server# bos create server.localdomain fs fs \
  /usr/afs/bin/fileserver /usr/afs/bin/volserver \
  /usr/afs/bin/salvager -noauth
```

Nastavíme uživatele serveru BOS tak, aby byl privilegovaný pro použití příkazů bos a vos, importujeme klíč který jsme si z Kerbera vyexportovali (pro import potřebujeme číslo klíče <kvno>, které získáme pomocí příkazu klist) a ověříme zda se v pořádku přidal:

```
server# bos adduser server.localdomain admin -noauth
server# asetkey add <kvno> /etc/openafs/afs.keytab afs/localdomain
server# bos listkeys server.localdomain -noauth
```

Nyní přidáme uživatele protection databáze a přiřadíme mu oprávnění:

```
server# pts createuser -name admin -noauth
server# pts adduser admin system:administrators -noauth
```

Pomocí této aplikace bychom později spravovali oprávnění jednotlivých klientů za použití skupin, zde je uživatel přiřazen do skupiny system:administrators, má tedy nejvyšší oprávnění.

V tuto chvíli můžeme server vypnout a zapnout ho již v režimu, kdy se bude nutné pro provádění některých příkazů autentizovat. To v našem případě jediného serveru zařídí přepínač -localauth. Druhým příkazem ověříme, zda všechny servery běží jak mají.

```
server# service openafs-fileserver stop
server# service openafs-fileserver start
server# bos status server -localauth
```

V případě nějakého neběžícího serveru nalezneme podrobnější informace ve výpisech chyb v adresáři /var/log/openafs. Chyba bývá většinou ve špatné synchronizaci času či špatném nastavením autentizace Kerberem.

Nyní je nutné vytvořit minimálně dva svazky, aby byl systém připojitelný. Prvním svazkem je root.afs, což je kořenový svazek, do kterého se vytvářejí další svazky. Druhým je root.cell, který se standardně vytváří na každém souborovém serveru. K veškeré manipulaci se svazky (synchronizace, vytváření, mazání, replikování atd.) se využívá aplikace s názvem vos. Před provedením těchto příkazů je nutné mít vytvořen adresář /vicepa, do kterého připojíme fyzický diskový oddíl naformátovaný jakýmkoli standardním souborovým systémem, například ext3. Pokud máme oddílů více, připojíme je do adresářů s názvy /vicepX (kde X nahradíme písmenem a až z). Tyto adresáře nemusíme nikde nastavovat, systém OpenAFS si je podle existujících názvů vybere sám, je tedy nutné dodržet pojmenování.

```
server# vos create server.localdomain vicepa root.afs -localauth
server# vos create server.localdomain vicepa root.cell -localauth
```

Nyní je systém připraven k připojení.

3.4.2 Klient

U klienta nainstalujeme balík:

```
klient# aptitude install openafs-client
```

Taktéž u klienta je nutný systém Kerberos s nastavením modulu PAM pro autentizaci oproti našemu nainstalovanému serveru. Také je potřeba klient synchronizace času, pro který poslouží buďto ntp-klient, nebo ruční synchronizace pomocí příkazu ntpdate.

Nastavíme seznam buněk, ke kterým se budeme chtít připojovat. Jedná se o konfigurační soubor /etc/openafs/CellServDB a již obsahuje rozsáhlý seznam veřejně dostupných serverů. Můžeme je ponechat a náš server přidat do tohoto seznamu v tomto tvaru:

```
>localdomain      # Cell name
192.168.1.1       # server
```

Nyní již můžeme zapnout klientský proces, tentokrát s přepínačem force-start, který za nás také nahraje požadovaný jaderný modul afs do paměti:

```
klient# service openafs-client force-start
```

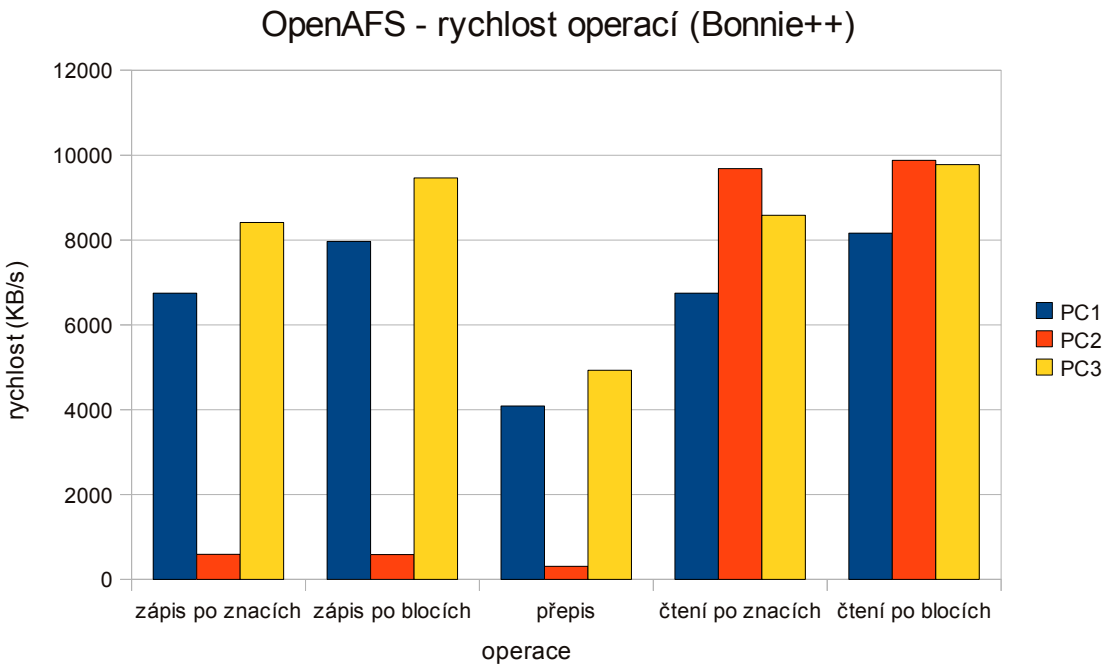
Od této chvíle se nám v adresáři /afs objeví seznam adresářů, které představují jednotlivé buňky ze seznamu CellServDB. V tuto chvíli jsou ale všechny pouze pro čtení. Abychom získali oprávnění zápisu do naší buňky, musíme se ověřit pomocí Kerbera. K tomu slouží následující příkazy:

```
klient$ kinit admin
klient$ aklog
klient$ tokens
```

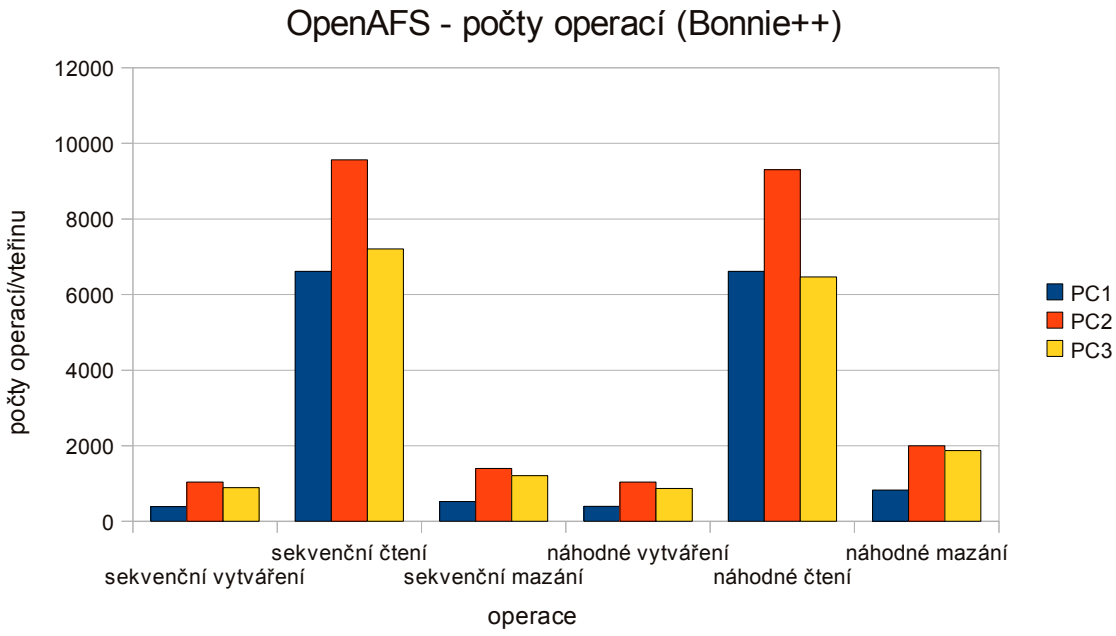
Prvním získáme tiket od Kerbera, po zadání správného hesla. Druhým příkazem získáme token důležitý pro ověření identity. Tokeny si můžeme vypsat posledním příkazem. Nyní jsme autentifikováni jako admin ze skupiny systém:administrators a máme právo zápisu do adresáře /afs/localdomain.

Připojený souborový systém pak máme možnost částečně řídit a nastavovat pomocí příkazu fs. Základními možnostmi je zjištění a nastavení kvót daných adresářů připojeného souboru a také jejich oprávnění (ACL). Dalších možností se zde nachází veliké množství, V případě potřeby je lze vyvolat příkazem fs help. [6], [15]

3.4.3 Benchmarky



Ilustrace 5: OpenAFS – rychlost operací



Ilustrace 6: OpenAFS – počty operací

3.4.4 Hodnocení

OpenAFS je pokročilým distribuovaným systémem s dlouhou historií. Celková instalace a konfigurace spolu s Kerberem je poměrně náročnější, ale k dispozici jsou dosti podrobné návody. V již zaběhnutých sítích jsou již systémy Kerberos (většinou spolu s LDAP) a NTP samozřejmostí pro potřeby ostatních služeb, tudíž by o to byla instalace OpenAFS jednodušší. Tento systém bude vyžadovat správu administrátorem, který se bude podle potřeb starat o jednotlivé svazky a provádět s nimi dostupné operace (migrování, klonování atd.).

Výkonové výsledky jsou přiměřené, pouze v případě druhé sestavy došlo k neidentifikovanému problému, nejspíše v oblasti autentizace, což způsobilo drastický propad rychlostí zápisu a přepisu až na 500 kilobajtů za sekundu. Tento problém se nepodařilo vyřešit. Na druhém grafu (Ilustrace 6) tento problém nezpůsobil žádný výkonový propad. Naopak systém vykazuje vysoký počet operací při čtení, což je důsledek vyrovnávací paměti DFS. Ostatní počty operací jsou průměrné.

3.5 Lustre

3.5.1 Instalace

Tento systém potřebuje pro svou funkčnost mimo samotných serverových aplikací také několik vlastností, které se ve standardním jádře Linux nenachází, proto jsou poskytovány záplaty, které tyto vlastnosti přidávají a upravují. V případě již zmíněných řešení enterprise je již záplatované jádro dodáváno jako součást distribuce, nebo je možné stáhnout dostupné balíky rpm. Takové se bohužel ale pro systémy Ubuntu nenabízí.

K dispozici jsou, jelikož se jedná o open source projekt, také zdrojové kódy. Ty jsou dokonce dostupné jako balík v distribučním repositáři. Je tedy možné aplikovat záplaty na vlastní jádro, což je ale proces náročný jak na čas (kompilace jádra může trvat až několik hodin), tak na znalosti jádra samotného.

Ač je tato problematika pravděpodobně nad rámec této práce, byly provedeny pokusy zkompilovat linuxové jádro verze 2.6.22 s dostupnými záplatami a jadernými moduly lustre. Kompilace proběhla v pořádku, ale jádro nebylo po instalaci schopno nastartovat, protože nastala chyba v připojování kořenového souborového systému. Příčinou mohla být

chyba v nastavení jádra před kompilací nebo pravděpodobněji nekompatibilita staršího jádra s novější verzí systému Ubuntu. Se starší verzí Ubuntu 8.04 LTS by teoreticky mohl být kompatibilní ale zkoumáno to nebylo. [8]

3.5.2 Hodnocení

Díky nezdařené kompilaci jádra nebylo možné pokračovat ve zprovoznění tohoto systému, proto ani nemohly být provedeny testy. Systém je dle referencí velice výkonný a široce používaný, bohužel jeho nasazení není na systémech Ubuntu bezproblémové. Bez výsledků ale nemůže být porovnáván s ostatními.

3.6 MooseFS

K dispozici jsou zdrojové kódy a balíky RPM odladěné pro některé distribuce. V Ubuntu repositářích je příliš stará verze, tudíž byla provedena kompilace ze zdrojových kódů.

Pro kompilaci je mimo standardních kompilačních nástrojů a knihoven potřeba balík libfuse-dev, pkg-config a zlib-dev. Archiv se zdrojovými kódy byl stažen z adresy moosefs.org/tl_files/mfscode/mfs-1.6.15.tar.gz.

Pokud skriptu configure nezadáme jinou volbu, bude provedena kompilace všech komponent (všech serverů i klientské aplikace). To pro testování vyhovuje, ale v případě nasazování tohoto systému by bylo vhodné vytvořit jednotlivé balíky pouze s potřebnými aplikacemi.

Po nainstalování zkompilované aplikace je umístění (pokud nezvolíme jiné pomocí configure) veškerých souborů v adresáři /usr/local/, logicky podle určení. Konfigurační skripty (přesněji řečeno jejich šablony) tedy nalezneme v adresáři /usr/local/etc/. Mají příponu cfg.dist, je tedy nutné těmto šablonám odstranit příponu dist a uvnitř těchto konfiguračních souborů odkomentovat veškeré potřebné nastavení.

3.6.1 Master server

Konfiguračními soubory tohoto hlavního serveru jsou mfsmaster.cfg a mfsexports.cfg. V prvním z nich je hlavní nastavení síťových portů, na kterých mezi sebou budou servery a klienti komunikovat. Druhý soubor slouží jako přístupový seznam. Tam nastavujeme roz-

sahy adres IP, které budou mít povoleno připojovat dané svazky a také jakým způsobem. Nastavuje se zde i přístup k meta svazku, který je pro nás důležitý tím, že se v něm nachází adresář odpadkového koše.

Před prvním spuštěním je třeba přejmenovat, nebo zkopírovat soubor metadata.mfs.empty na metadata.mfs v adresáři /usr/local/var/mfs. Server pak spouštíme a ovládáme příkazem mfsmaster s parametrem [start|stop|restart|reload].

Na master serveru je dobré spustit také přístup ke sledovacímu skriptu cgi dostupnému pak jednoduše z webového prohlížeče. Spouští se příkazem mfscgiserv bez jakýchkoli parametrů.

Druh komunikace	**port**
Master ↔ metalogger servery	9419
Master ↔ chunk servery	9420
Master ↔ klient	9421
Chunk server ↔ klient	9422
CGI server ↔ klient	9425

Tabulka 1: MooseFS standardní TCP komunikační porty

3.6.2 Chunk server(y)

Hlavní konfigurace se provádí v souboru mfschunkserver.cfg, kde nastavujeme především doménové jméno masterserveru, ke kterému se bude připojovat, a jeho port. Dále pak port, na kterém bude komunikovat s klienty. Další možnosti je možné najít v manuálových stránkách, ale pro testování postačily přednastavené hodnoty.

Konfigurační skript mfshdd.cfg obsahuje, po řádcích zadané, přípojné body diskových oddílů, kam si bude ukládat data. Ze zřejmých důvodů není doporučováno využívat lokálního souborového systému, na kterém běží systém. V případě nouze je zde možnost vytvoření souboru o určité velikosti, ten naformátovat běžným FS a poté jej připojit do daného přípojného bodu.

Chunk server nastartujeme pomocí příkazu:

```
server# mfschunkserver start
```

3.6.3 Klient

Vzdálený oddíl připojíme pomocí příkazu mfsmount:

```
klient$ mfsmount mfsmountpoint -H server
```

Na již připojeném systému můžeme nastavit čas, po který se nebude fyzicky obsah zadaných složek/souborů mazat, ale ponechá se v koši. Nastavení se provádí příkazem mfssettrashtime a nastavený čas můžeme zjistit příkazem mfsgettrashtime.

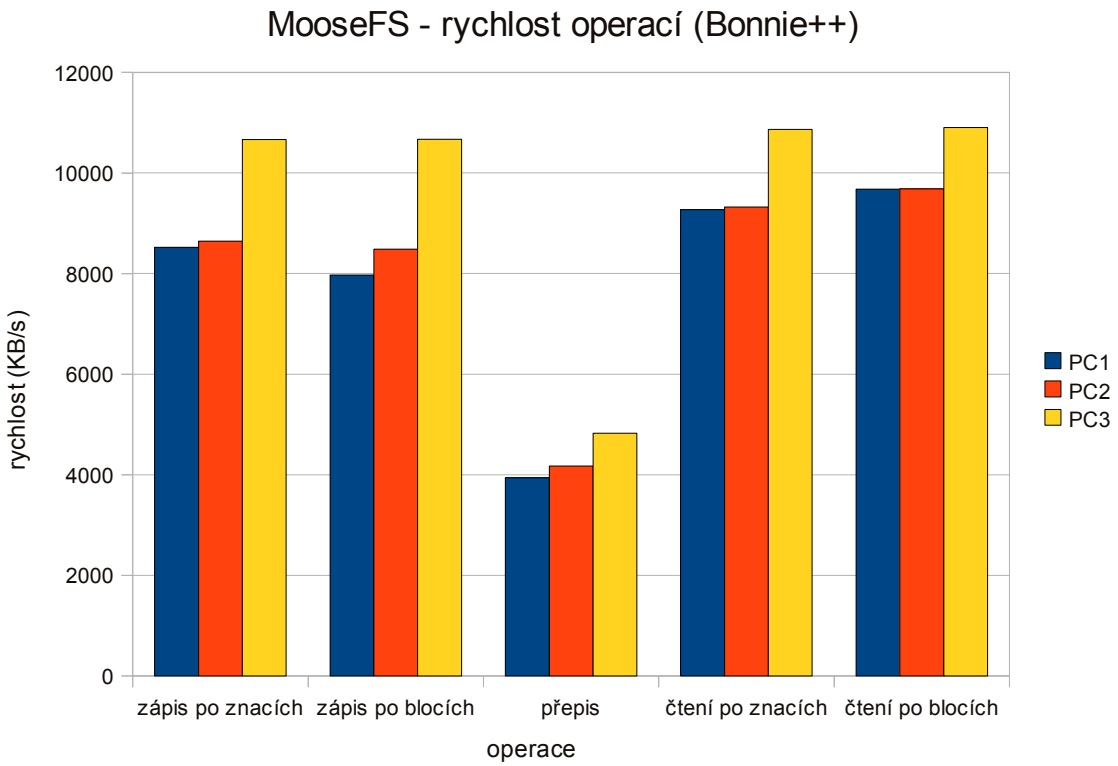
Podobně můžeme nastavovat, v kolika kopiích se má soubor/adresář ukládat (příkazy mfsgetgoal a mfssetgoal). Soubor se v tom případě pokusí uložit na daný počet chunk serverů. Pokud jich tolik není dostupných, **neukládá** se vícekrát na stejný. Pokud ale za chodu přidáme další chunkserver, soubor se automaticky replikuje.

Dalším příkazem je mfssnapshot, ten nám vytvoří snapshot (aktuální stav) souboru nebo adresáře. Takovýto snapshot nezabírá žádné místo navíc do té doby, než dojde ke změně původního souboru nebo adresáře. v tom případě zachová původní data.

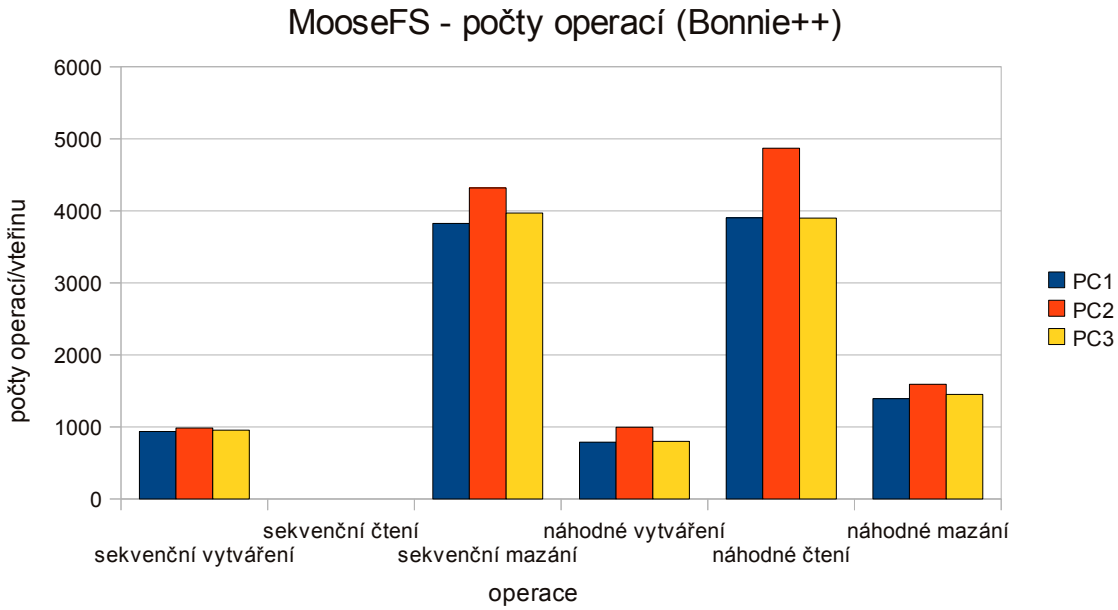
3.6.4 Obecně

Při zapínání tohoto systému je doporučováno spouštět procesy v pořadí masterserver, mfsmetaloggerservery, chunkservery, klienti. Pro vypínání platí samozřejmě opačné pořadí. [9]

3.6.5 Benchmarky



Ilustrace 7: MooseFS – rychlost operací



Ilustrace 8: MooseFS – počty operací

3.6.6 Hodnocení

Systém se sice musí kompilovat ze zdrojových kódů, ale to není v tomto případě žádný problém. Kompilace proběhla rychle a bez problémů. Konfigurace je také poměrně jednoduchá a od administrátora nebude vyžadovat žádnou speciální pravidelnou správu, navíc systém obsahuje přehledné informativní webové rozhraní.

Výkonově je na tom tento systém také výborně, nedosahuje sice nejvyšších hodnot, ale výsledky jsou vyrovnané. V počtech operací za vteřinu je pak tento DFS pod průměrem v případě vytváření, naopak v ostatních operacích je nadprůměrný, přičemž při sekvenčním čtení dosáhl opět neměřitelných (vysokých) hodnot způsobených vyrovnávací pamětí.

3.7 CODA

K dispozici jsou zdrojové kódy, binární instalátor pro Linux, FreeBSD a Microsoft Windows NT. Pro linuxové distribuce navíc ještě repositáře s balíčky typu deb a rpm. Nejlepší variantou bývá většinou repositář, jelikož se vždy dozvíme o nových verzích a systém nám sám nabídne případné aktualizace.

3.7.1 Server

Přidáme repositář do /etc/apt/sources.list, máme možnost stable, testing nebo unstable. V době psaní práce ale obsahují stejné verze.

```
deb http://www.coda.cs.cmu.edu/debian stable/
```

Instalace balíku coda-server, aptitude přidá automaticky jeho závislosti (coda-update, liblwp2, librpc2-5, librvm1 a rvm-tools), nejaktuálnější verze k dispozici byla 6.9.4.

```
server# apt-get update
server# apt-get install coda-server
```

Spuštění hlavního interaktivního skriptu /usr/sbin/vice-setup, který za nás nastaví veškeré další konfigurační skripty. Tento skript se smí spustit pouze jednou, jinak způsobí nefunkčnost konfigurace. V případě, že jsme něco pokazili a potřebujeme spustit konfiguraci znovu, je nutné vypnout serverové procesy (volutil shutdown; pkill updateclnt; pkill updatesrv; pkill auth2) a potom smazat adresář /vice.

Vice-setup se bude postupně ptát na otázky ohledně konfigurace (v hranatých závorkách přednastavené hodnoty hodnoty):

```
Do you want the file /etc/coda/server.conf created? [yes]
```

Vytvořit konfigurační soubor?

```
What is the root directory for your coda server(s )? [/vice]
```

Adresář pro odkládací soubory, logy, databázové soubory apod.

```
Is this the master server, aka the SCM machine? (y/n)
```

Je toto master server? Musí být alespoň jeden.

```
Enter a random token for update authentication: updatetoken
The following token must be identical on all servers.
```

Nastavení tří klíčů (pro update, auth2 a volutil), které se budou používat mezi coda servery pro ověření identity. Musí být tedy stejné pro všechny servery které spolu mají komunikovat.

```
Enter an id for the SCM server.
```

Nastavení ID serveru. Musí to být číslo v rozmezí 0–255, přičemž by se neměla použít čísla 0, 127 a 255.

```
Enter the uid of this user:
Enter the username of this user
```

Skript vytvoří uživatele, který bude použit pro administraci serveru coda. Tento uživatel není systémový, používá se pouze pro autorizaci s coda. Je to obdoba uživatele vytvářeného u systému OpenAFS ve službě Kerberos.

```
Are you ready to set up RVM? [yes/no]
```

Jsme připraveni pokračovat v nastavení RVM?

```
What will be your log file (or partition)? /usr/local/coda/RVM_LOG
```

LOG se používá pro uložení transakcí před provedením.

```
What is your log size? (enter as e.g. '20M') 20M
Where is your data file (or partition)? /usr/local/coda/RVM_DATA
```

V tomto souboru se budou ukládat metadata pro naše data. Logicky to znamená to, že pokud budeme skladovat veliké soubory, budeme potřebovat daleko menší soubor s metadaty, než když budeme ukládat veliké množství malých souborů. Pro předběžný odhad je

tu aplikace rvmsizer, která požadované místo vypočítá z daného adresáře. Pro představu „#rvmsizer /“ na čerstvé instalaci Ubuntu 9.10 server (500 MiB, 40 000 souborů, 6 000 adresářů) vypočítá zhruba 23 MiB. Je to ale pouze příklad, který navíc obsahuje spoustu odkazů a virtuálních souborových systémů generovaných jádrem.

```
What is the size of you data file (or partition)
[32M, 64M, 128M, 256M, 512M, 768M, 1G]: 256M
```

Velikost souboru pro metadata, musíme si uvědomit, že tento soubor bude celý mapován do virtuální paměti z důvodu rychlosti, navíc by nebylo dobré kdyby byl příliš odkládán do swapu.

```
Proceed, and wipe out old data? [y/n]
```

Pokračovat a vyčistit (vynulovat) právě nastavené soubory?

```
Where shall we store your file data [/vicepa]?
```

Kam se budou ukládat již skutečná data na coda serveru?

```
Select the maximum number of files for the server. [256K, 1M, 2M,
16M]:
```

Maximální počet souborů v souborovém systému.

```
Shall I try to get things started? (y/n)
```

Pokud jsme vše nastavili správně, může se pro nás setup zkusit nastartovat všechny potřebné služby. Pro případ manuálního startu:

```
server# auth2 &
server# updatesrv
server# updateclnt -h server.localdomain
server# startserver
```

Server.localdomain je doménové jméno, pod kterým bude server coda dostupný na síti.

V případě, že se automatický start provedl bez chyby, vice-setup se pro nás pokusí vytvořit počáteční kořenový svazek. Manuálně bychom jej vytvořili takto:

```
server# createvol_rep / ubuntu-oliwa/vicepa
```

Podobně, pouze s jiným názvem, bychom vytvářeli další svazky, například pro domovské adresáře uživatelů.

3.7.2 Klient

Tak jako u serveru přidáme stejný repositář a nainstalujeme balík coda-client s automaticky přidanými závislostmi:

```
klient# aptitude install coda-client
```

Spustíme interaktivní konfigurační skript, který se nás zeptá na velikost vyrovnávací paměti (není doporučováno více jak 300 MiB) a na standardní autentizační doménu. Skript pak zajistí vytvoření zařízení /dev/cfs0 pro komunikaci s jádrem, vytvoří potřebné adresáře a soubory a inicializuje soubory vyrovnávací paměti.

```
klient# venus-config server 102400
```

Nyní můžeme spustit server (první spuštění se provede s přepínačem init):

```
klient# venus -init
```

Pokud start proběhl v pořádku, měli bychom mít přístup do adresáře /coda/localdomain, nyní pouze pro čtení. Pro vyšší oprávnění se musíme, stejně jako u systému OpenAFS, za pomoci Kerbera autentizovat pomocí aplikace clog.

```
klient$ clog user@localdomain
```

Následně máme možnost připojený systém částečně ovládat pomocí příkazu cfs, jehož nápovědu vyvoláme přepínačem help. [16]

3.7.3 Benchmarky

Ač se povedlo provést instalaci a konfiguraci tohoto systému bez problému a systém byl funkční (daly se s ním provádět běžné operace), nasazení testovacích nástrojů tento systém nezvládl. Každý test způsobil po určité době kompletní zaseknutí klientského procesu, který se pak musel násilně ukončit a byl problém ho znovu nastartovat. Nelze s jistotou říci, zda se jednalo o ojedinělou chybu v dané verzi aplikace v konkrétní konfiguraci nebo zda tento DFS prostě nezvládá tak vysokou zátěž, která na něj byla kladena při testech.

3.7.4 Hodnocení

Tento DFS je založen na systému AFS, implementuje si oproti němu ověřování a časovou synchronizaci sám, čímž usnadňuje instalaci, konfiguraci a následnou správu. Napomáhají také prvotní interaktivní skripty, které pro nás nastaví spoustu konfiguračních souborů a vytvoří vše potřebné. Bez testů ale bohužel nelze porovnat výkonové výsledky.

3.8 TahoeLAFS

Tento celý systém je napsán v programovacím jazyce Python, pro běh je tedy potřeba mít v systému python verze 2.4, 2.5 nebo 2.6 (v3 prozatím nelze).

K dispozici jsou zdrojové kódy s instalátorem, který pouze nakopíruje skripty na potřebná místa. Jazyk python je totiž interpretovaný a nepotřebuje zdrojový kód kompilovat. Ke stažení jsou také předpřipravené deb balíky pro jednotlivé distribuce Debian/Ubuntu, v Ubuntu od verze 9.10 je dokonce balík přítomen přímo v distribučním repositáři.

3.8.1 Instalace

Nejjednodušší je tedy nainstalovat balík z repositáře, k němu se přidá automaticky několik závislých pythonových knihoven:

```
# aptitude install tahoe-lafs
```

Tento balík instalujeme na všech zúčastněných počítačích, obsahuje totiž universální aplikaci pojmenovanou tahoe, pomocí které se provádějí veškeré operace.

Nejprve vytvoříme prázdný adresář. Poté na tomto adresáři zavoláme vytvoření introduceru a následně ho spustíme:

```
introducer# mkdir /introducer
introducer# tahoe create-introducer -C /introducer
introducer# tahoe start /introducer
```

Tím se nám v daném adresáři vygenerovala spousta konfiguračních a pomocných souborů. Pro další postup je pro nás důležitý hlavně soubor introducer.furl který budeme kopírovat každému klientovi.

Souborový server nastartujeme tak, že vytvoříme a pustíme klienta. Klient má totiž v základním nastavení povoleno ukládání dat. To se nastavuje v konfiguračním souboru

tahoe.cfg. Inicializujeme tedy klienta, automaticky se vytvoří adresář tahoe v domácím adresáři uživatele. Sem také nakopírujeme zmíněný soubor introducer.furl a klienta zapneme:

```
introducer$ tahoe create-client
introducer# cp /introducer/introducer.furl /home/user/.tahoe/
introducer$ tahoe start
```

Nyní úplně stejně zprovozníme klientský proces na počítači klienta, pouze v konfiguračním souboru /home/user/.tahoe/tahoe.cfg nastavíme:

```
[storage]
enabled = false
```

Tím máme zprovozněn celý základní systém, který by měl fungovat. Systém umožňuje několik způsobů ovládání: webové rozhraní, příkazová řádka, modul FUSE a další.

Pro testování je potřeba daný systém připojit do lokálního souborového systému, což by měl umožnit právě modul FUSE. Ten je k dispozici ve třech implementacích, bohužel ani jedna z nich se nepovedla zprovoznit. Vypadá to, že tyto implementace byly vytvořeny pouze jako doplněk nějakými uživateli, nejsou udržovány a obsahují chyby.

Díky nemožnosti připojit lokálně souborový systém, nebylo možné provést benchmarky. Byly tedy alespoň vyzkoušeny možnosti za použití webového a příkazového rozhraní.

Webové rozhraní se nachází na lokální adrese http://127.0.0.1:3456/. V informační oblasti se dozvíme služby běžící na našem klientském stroji a k jakým souborovým serverům jsme připojeni. Dále pak máme možnost vytvářet adresáře, nahrávat a získávat soubory a nastavovat jim dodatečné atributy.

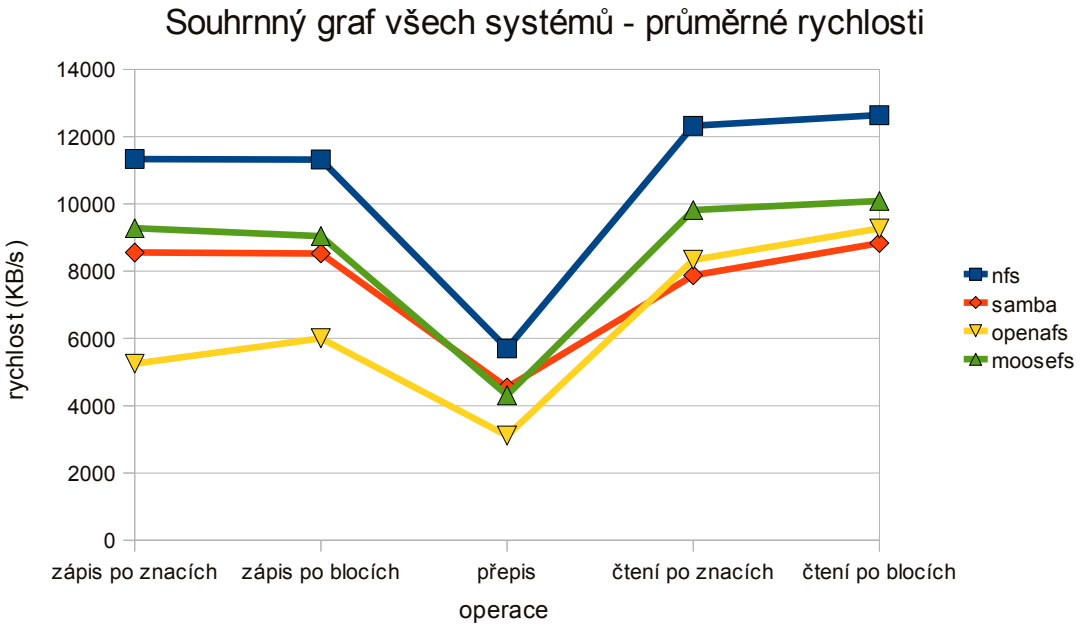
Příkazové rozhraní se ovládá také pomocí příkazu tahoe. Máme možnost pomocí parametrů tohoto příkazu provádět základní operace vytváření, mazání a přejmenování adresářů a souborů. Výpis dostupných parametrů je možné provést příkazem tahoe –help. [11]

3.8.2 Hodnocení

Bez provedených testů nelze tedy tento systém hodnotit z výkonového hlediska. Lze pouze říci, že je tento systém velice jednoduché nainstalovat a nakonfigurovat. Použití je taktéž poměrně jednoduché, ale uživatelé se oproti ostatním souborovým systémům musí naučit používat některé z dostupných rozhraní.

Závěr

Úkolem bylo zprovoznit a otestovat co nejvíce DFS. Po prozkoumání dostupných systémů bylo vybráno sedm z nich. Z nich se podařilo úspěšně nainstalovat a nakonfigurovat šest a následně otestovat se povedlo pouze čtyři systémy.



Ilustrace 9: Souhrnný graf všech systémů

Z pohledu maximálního výkonu lze tedy dle výsledků doporučit systém NFS, který dosahoval nejvyššího průměrného výkonu při všech operacích, jak lze vidět na souhrnném grafu (Ilustrace 9). Druhého celkově nejlepšího výsledku dosáhl systém MooseFS s průměrným rozdílem dvou megabajtů za sekundu oproti NFS.

Po porovnání všech výsledků všech třech sestav lze konstatovat, že prvkem ovlivňujícím výkonnost těchto systémů v testovaném prostředí nebyl výpočetní ani diskový výkon počítačových sestav, ale propustnost sítě. Potvrzují to poměrně vyrovnané výkony na všech sestavách. Přitom PC1 byl osazen oproti PC3 několikanásobně pomalejším procesorem, méně operační paměti a také diskem s pomalejšími otáčkami.

Je ale třeba mít na paměti, že testy byly provedeny v jeden čas pouze jedním klientem a s pouze jedním serverem. Pokročilejší testované systémy tedy nedostaly příležitost prokázat svůj výkon v prostředí, pro které jsou připraveny a v kterém jsou široce nasazovány. Při

vyhranění vlastních fyzických serverů pro metadata a nasazení více datových serverů u systémů, které toto podporují, by výsledky byly několikanásobně lepší v jejich prospěch.

Pro hlubší porovnání a více vypovídající hodnocení by bylo nutné provést testy postupně s více fyzickými servery a také s více klienty. Testování by ale bylo nesmírně časově náročné, protože již testy prováděné v této práci vyžadovaly desítky až stovky hodin testů.

Základní pojmy a zkratky

FS – souborový systém

DFS – distribuovaný souborový systém

metadata – dodatečné atributy pro data (název, vlastník, čas vytvoření, …)

cluster – skupina spolupracujících počítačů spojených sítí

ACL – seznam pro řízení přístupu, oprávnění

POSIX – standardizované přenosné rozhraní operačního systému

GNU GPL – nejběžnější licence pro svobodný software

RPC – vzdálené volání procedur

DKMS – systém pro dynamickou kompilaci a nahrávání jaderných modulů

PAM – univerzální systém pro aplikačně nezávislé autentizaci pomocí modulů a schémat

Seznam ilustrací

Použitá literatura

[1] MENASCÉ, Daniel A. *Distributed File Systems: Part I* [online]. 30. 11. 1998 [cit. 2010-05-03]. Distributed File Systems. Dostupné z WWW: <http://www.cs.gmu.edu/~menasce/osbook/distfs/index.html>.

[2] List of file systems. In *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida): Wikipedia Foundation, 29. 4. 2010 [cit. 2010-05-03]. Dostupné z WWW: <http://en.wikipedia.org/wiki/List_of_file_systems#Distributed_file_systems>.

[3] NOWICKI, Bill. *RFC1094 – NFS: Network File System Protocol Specification* [online]. Sun Microsystems, březen 1989 [cit. 2010-05-03]. Dostupné z WWW: <http://tools.ietf.org/html/rfc1094>.

[4] *Samba: Opening Windows to a Wider World* [online]. 2010 [cit. 2010-05-03]. Dostupné z WWW: <http://www.samba.org>.

[5] OpenAFS. In *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida): Wikipedia Foundation, 27. 12. 2009 [cit. 2010-05-03]. Dostupné z WWW: <http://en.wikipedia.org/wiki/OpenAFS>.

[6] *OpenAFS: Documentation* [online]. 28. 8. 2009 [cit. 2010-05-03]. Dostupné z WWW: <http://docs.openafs.org/index.html>.

[7] Lustre (file system). In *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida): Wikipedia Foundation, 23. 4. 2010 [cit. 2010-05-03]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Lustre_(file_system)>.

[8] *Lustre: High Performance and Scalability* [online]. Oracle, c2010 [cit. 2010-05-03]. Dostupné z WWW: <http://wiki.lustre.org/>.

[9] *MooseFS: A file system for highly reliable petabyte storage* [online]. Gemius, c2002 [cit. 2010-05-03]. Dostupné z WWW: <http://www.moosefs.org/>.

[10] Coda (file systém). In *Wikipedia: the free encyclopedia* [online]. St. Petersburg (Florida): Wikipedia Foundation, 6. 2. 2010 [cit. 2010-05-03]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Coda_(file_system)>.

[11] *Tahoe-LAFS: Documentation for Tahoe Users* [online]. Allmydata, 2010-05-04 [cit. 2010-05-04]. Dostupné z WWW: <http://allmydata.org/trac/tahoe-lafs/wiki/Doc>.

[12] MARTIN, Ben. *Using Bonnie++ for filesystem performance benchmarking* [online]. Linux.com, 1. 7. 2008 [cit. 2010-05-03].. Dostupné z WWW: <http://www.linux.com/archive/feature/139742>.

[13] *NFSv4Howto* [online]. Canonical, 2010 [cit. 2010-05-03]. Ubuntu Documentation: Community Documentation. Dostupné z WWW: <https://help.ubuntu.com/community/NFSv4Howto>.

[14] *Samba* [online]. Ubuntu Česko, 2009-09-22 [cit. 2010-05-03]. Dostupné z WWW: <http://wiki.ubuntu.cz/Samba>.

[15] BOND, Forest. *Kerberos On Ubuntu* [online]. c2005–2010 [cit. 2010-05-05]. Dostupné z WWW: <http://www.alittletooquiet.net/text/kerberos-on-ubuntu/>.

[16] BRAAM, Peter, et al. *Coda File System: The Coda HOWTO* [online]. Verze 1.01. 16. 1. 2000 [cit. 2010-05-03]. Dostupné z WWW: <http://www.coda.cs.cmu.edu/doc/html/coda-howto.html>.

[17] KRČMÁŘ, Petr. Nebojíme se kompilace – II (praxe). In *AbcLinuxu* [online]. Stickfish, 19. 10. 2004 [cit. 2010-05-13]. Dostupné z WWW: <http://www.abclinuxu.cz/clanky/navody/nebojime-se-kompilace-ii-praxe>. ISSN 1214-1267.

Příloha – Obsah CD

- Tato práce ve formátu pdf
- Soubor „provedené testy.ods“ otevřeného datového typu ODF, obsahuje veškeré naměřené hodnoty a grafy