Samostatná práce z předmětu X36MTI

► Rsync ◄

Vypracoval: **Michal Fiala**
Skupina: Pa 07:30
Kontakt: fiala@mfiala.net

Obsah

1. Cíl práce

V této práci se pokusím popsat nástroj rsync. Bude mě zajímat, jak algoritmus rsync funguje, ale budu také klást důraz na praktické využití nástroje rsync. Pokusím se popsat nesporné výhody i nevýhody používání nástroje rsync z pohledu unix* administrátora.

2. Historie a účel rsync

Rsync je platformě nezávislý nástroj pro synchronizace datových uložišť. Tento nástroj využívá alogirtmu taktéž nazvaného rsync, na jehož návrhu se především podílel australský kodér Andrew Tridgell a na jehož funkčnosti je rsync postaven. Prvotním záměrem rsync bylo minimalizovat datový tok při synchronizaci, tedy nálezt takové řešení, které šetří přenosové pásmo. Přenosovým pásmem můžeme chápat interní sběrnci systému ale také jakákoliv síťová linka, na které běhá rodina protokolů TCP/IP. Původně je synchronizace postavena na síťovem modelu klient-server, kdy server naslouchá na TCP portu 873.

3. Jádro algoritmu rsync

Algoritmus rsync řeší následující problém.

Představme si, že mám dva soubory A a B. Chceme B zaktualizovat tak, aby odpovídal A. První věc, která nás napadne, je zkopírovat A na místo B. Jednoduché a účinné řešení v případě, kdy data nepřenášsíme po velice pomalé lince – soubory v řádu MiB a linka o přenosové rychlosti v řádech kib/s. Částečným řešením může být soubor A zkomprimovat a přenést po lince, zrychlení je však pouze v řádu jednotek a pro veliké data tedy nevyhovuje.
Nyní uvažujme, že jsou si oba soubory dosti podobné, řekněme že původně vycházely ze stejného, ale postupem času se jejich obsah měni různým způsobem. Řešením by mohlo být zjištění rozdílu mezi A a B a rozdíl aplikovat na B. Zásadní je však, že ke zjištění rozdílu A a B je zapotřebí znát obsah obou souborů a my víme, že soubory jsou umístěny na vzdálených uložištích. A přesně tento problém rsync řeší – rsync efektivně zjistí, které části souboru A je zapotřebí přenést tak, aby se soubor B mohl zaktualizovat do podoby souboru A.

Rekapitulace

- stroj α drži soubor A
- stroj β drži soubor B
- A a B jsou podobné soubory
- mezi α a β je pomalá linka
- soubor B chceme zaktualizovat dle souboru A

Průběh algoritmu

1. β rozdělí soubor B do série neopakujících se datových bloků o kosntatní velikosti S. Poslední blok může být menší nežli S
2. Pro každy z bloků β vypočítá 2 hashe
   silný hash - 128b
   slabý rotující hash - 32b
3. β pošle všechny hashe straně α
4. α se pokusí nálezt v celém A bloky specifikovné zaslanými hashi (o jekémkoliv posunutí, né pouze o posunutí S). Tato operace proběhne rychle díky důmyslnoti rotujícího hashe a především během pouze jednoho průchodu.
   (blizsi info http://rsync.samba.org/tech_report/node3.html)

5. α posílá β sekvencí instrukcí jak sestavit soubor B. Každý instrukce je buď ukazatelem na existující blok dat v B nebo data. Data se posílají pouze v případě, že B tyto data neobsahuje.
6. β nyní disponuje instrukcemi dle kterých lze poskládat soubor B tak, aby odpovídal souboru A. Potřebné informace získal především pouze během jednoho výměného cyklu (roundtrip), což jenom potvrzuje šetrný přístup algoritmu k propustnosti přenosové linky.

4. Funkčnost rsync jako celku

V 3. kapitole jsem si popsali, jak funguje efektivní synchronizace dvou souborů. Nyní si popíšeme funkčnost nástroje rsync jako celku.

Nejdříve ujasním role v systému:

client	inicializuuje, startuje synchronizaci
server	vzdálený rsync proces nebo systém, ke kterému se client připojuje (lokální přenos, nebo pomocí vzdáleného shellu, nebo pomocí síťového socketu) např. rsyncd, sshd, ...
démon	rsync démon, který čeká na příchozí požadavky od rsync klientů
vzdálený shell	jeden nebo více procesů, které zprostředkovávají komunikaci mezi rsync klientem na straně jedné a rsync démonem na straně druhé
vysílač	rsync proces, který má přístup ke zdrojovému souboru (vzor)
prijímač	rsync proces, který má přístup k cílovému uložisti, uložisti, jež chceme zesynchronizovat (přijímá instrukce od vysílače a zapisuje aktualizace na disk)
generátor	proces, který zastřešuje souborovou logiku, identifikuje změněné soubory

Celý postup synchronizace stromu uložiště lze rozdělit do několika fázi:

1. **Inicializace**
   Jakmile je spuštěn rsync klient s duchaplnými parametry, provede spojení se serverem. Toto spojení může být provedené pomocí roury (pipe) nebo síťové sockety.
   Roura je použita v případě, kdy je spojení navazováno pomocí vzdáleného shellu, např. využiti ssh – na straně serveru se spustí vzdálená shell, který spustí rsyncd s příslušnými parametry. Vysílač a přijímač spolu komunikují pomocí vzdáleného shellu skrz rouru.
   Roura se také použije v případě, kdy dochazi k synchronizaci pouze v rámci lokalních uložišť, klient provede fork, vytvořený proces se stavá serverem.
   Síťový socket je použit v případě, kdy je spojení navazováno přimo s démonem.
   Následuje vzájemná domluva na verzi rsync protokolu, je zvolena nejmenší hodnota verze.

2. **Seznam souborů**
   Seznam souborů obsahuje cestu k souborům, vlastnictví uživatele a skupiny, práva, velikost, čas poslední modifikace. Pokud je navíc specifikován i parametr –checksum, obsahuje seznam i hash.
   První věc, kterou vysílač provede, je sestavení seznamu souborů. Během sestavováni seznamu, posílá vysílač každou položku i přijímači. Jakmile je seznam přenesen, setřídí se dle cesty k souboru a tabulka se opatří indexy. Indexy poslouži pro další komunikaci mezi vysílačem a přijímačem.
   Přijímač provede fork a z druhého vzniklého procesu se stane generátor.

3. **Generátor**
   Generátor pracuje se soubory na straně přijimáče, porovnává a provádí příslušnou činnost. Pokud je definovaný parametr –delete, identifikuje soubory neexistující na straně vysílače a odstraní je. Sekvenčně prochází seznam souborů a zjišťuje, zdali je možné soubor přeskočit, tedy neprovádět kompletní algoritmus rsync nad tímto souborem. Výchozí nastavení říký, že se kontroluje pouze čas modifikace a velikost souboru. Pokud navíc klient specifikuje parametr –checksum, potom je do výběrové operace zahrnuta operace hashe. Automaticky se nepřeskakují adresáře, soubory zařízení, symbolické linky. Neexistující adresáře jsou vytvořeny. Pokud soubor nebylo možné přeskočit, stává se z něj tzv. základní soubor, který slouži jako základ pro vytvoření hashů. Generátor předá vytvořené hashe následované indexem souboru přijímači a ten je odešle vysílači. Toto už je operace blíže popsaná ve 3. kapitole.

4. **Vysílač**
   Provádi se algoritmus popsaný ve 2. kapitole. Pouze stručně. Vysílač nalezne dle indexu synchronizovaný soubor, pokusí se nálezt v celém souboru bloky specifikovné zaslanými hashi (o jekémkoliv posunutí, né pouze o posunutí S). Vysílač zašle přijímači sekvencí instrukcí jak sestavit synchronizovaný soubor. Každý instrukce je buď ukazatelem na existující blok dat v bázovém souboru nebo data. Na konci komunikace v rámci jednoho souboru vysílač provedy hash celého souboru a zašle přijímači. Na straně vysílače se kladou zvýšené nároky na výpočetní výkon, vzhledem k nutnosti často počítat jednosměrné filtry.

5. **Přijímač**
   Provádí se algoritmus popsaný v 2. kapitole. Přijímač obdrží instrukce od vysílače, jak sestavit synchronizovaný soubor. Otevře bázový soubor ale také vytvoří dočasný soubor. Nenalezené bloky zapisuje na určené místo do dočasného souboru, nalezené bloky přemisťuje na správne místo do do časného souboru. Jakmile je dočasný soubor kompletně dokončen, zkontroluje se celkový hash. Pokud neodpovídá, dočasný soubor je odstraněn a celá operace se opětovně opakuje ještě jednou. Pokud po opakované synchronizaci bazového souboru opět došlo k chybě, je nahlášena chyba. V opračném případě se změní vlastníctví, práva, čas modifikace dočasného souboru a přepíše se bázový soubor.
   Na straně přijímače se kladou zvýšené nároky na IO operaci při vytvýření dočasných souborů, tato skutečnost je především markatní u velkých souborů, u malých souborů se dobře uplatní cache.

5. Rsync pro a proti

Rsync je velice mocný nástroj pro práci se soubory, kompletní synchronizace uložišt je s jeho pomocí otázkou spravné volby několika parametrů. Uveďme si jeho klady, ale i zápory.

Klady

- optimalizace velikosti přenášených dat (rsync protokol, komprese), neocenitelné na pomalejších linkách
- kompletní synchronizace uložiště různou formou
  - vlastníctví
  - práva
  - čas modifikace
  - zachovaní symbolických, hard linků
  - přenos speciálních souborů (zařízení)
- kontrola integrity přenášených dat
- vysoka efektivita díky protokolu s nízkou režií

- nástroj rsync je výtečně konfigurovatelný – mocný nástroj při použiti zkušeným administrátorem

Zápory

Pro dobře navržený komunikační protokol by mělo dle konvencí platit

- veškerá data jsou posílana v dobře formovaných paketech s hlavičkou, tělem
- hlavička jasně definuje a popisuje obsah těla, tedy např i délku
- pro některé protokoly platí vlastnosti jako opětovné sestavení porušeného spojení, dobře čitelné, vzáhjemna paketová nezávislost, bezestavová komunikace, apod

Pro protokol rsync neplatí žadný z výše uvedených rysů, data jsou zasílána v neporušených bytových proudech. Význam dat je definován v ramci kontextu komunikace. Tento způsob komunikace dobře funguje za spolehlivých podmínek, ovšem je velice špatně dokumntovatelný, sledovatelný či rozšířitelný.

Jako další zápory protokolu rsync bychom mohli zmínit:

- vysoká náročnost na množství operační paměti na straně vysílače i přijímače v případě, kdy se synchronizuje uložiště s velkým počtem souborů (seznam souborů nabýva velkých hodnot – průměrně 100B na položku) – vývojáři tvrdí, že některá z následujících verzí rsync by měla optimalizovat paměťovou nenasytnost rsync nástroje
- vysoká náročnost na výpočetní výkon na straně vysílače, jelikož se dopočítává velké množství hashů
- vysoká náročnost na IO operace na straně přijímače, jelikož je nutné pro každy bázový soubor vytvořit dočasný soubor a provádět nad ním operace

6. Rsync v praxi

Rsync obsahuje celou řadu parametrů, pokusím se přiblížit ty nejvíce používané. Obecný zápis příkazu rsync

```
rsync [OPTION]... SRC [SRC]... DEST
rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST
rsync [OPTION]... SRC [SRC]... [USER@]HOST::DEST
rsync [OPTION]... SRC [SRC]... rsync://[USER@]HOST[:PORT]/DEST
```

kde OPTION specifikuje volby, SRC specifikuje zdroj dat, DEST specifikuje cíl synchronizace, USER uživatelský účet na vzdýleném systému, HOST stroj na kterém jsou uložená DEST data, PORT porta na kterém poslouchá rsync démon.

Rozebereme si několik příkladů použití

Inkrementální synchronizace v rámci jednoho systému

```
rsync -av /src/foo /dest
```

Základní inckrementální synchronizace cíle /dest se zdrojem /src/foo. Vzhledem k tomu, že /src/foo nekončí lomítkem '/', bude se synchronizovat i samotný adresáč foo, néjen jeho obsah.
Parametr -a (--archive) je zásadní, je to alias pro parametry -rlptgoD, pojďme se na ně podívat blíže

- **r** (- --recursive) - rekurzivně
- **l** (--links) – kopíruj symbolické odkazy jako symbolicé odkazy, nikoliv jejich cíle
- **p** (--perms) – nastav přístupová práva
- **t** (--times) – nastav čas modifikace
- **g** (--group) – nastav vlastnictví pro skupinu

- **o** (--owner) – nastav vlastnictví
- **D** (--devices –specials) – přesuň speciální soubory (tzn. i soubory blokových, znakových zařízení)

Paramtr **-v** není nic jiného než –verbose, tedy upovídaný výstup. Pokud Vás zajímá více o průběhu opearace, určite uvítate parametr –**progress.**
Tento příklad je tedy značne vzorový, paraemtr **-a** symbolizuje reflektuji nejčastější požadavky na inkrementální synchronizaci.

Synchronizace mezi dvěma hosty

```
rsync -avz --progress --delete /src/foo/ hostname::modulname/dest
```

Úplná synchronizace mezi dvěma uložisti na dvou různých strojích. Neznámé parametry jsou pro nás

- **z** (--compress) – použuj kompresi při přenosu, tímto se dá dále ušetřit šířka přenosové pásmo
- **--delete** – odstraň na straně přijímače souboury, které se nenalézají na straně vysílače (lze přesně specifikovat kdy parametry –delete-befor, --delete-during, --delete-after)

Všimněte si, že /src/foo končí lomítkem '/' , což znamená že se synchronizuje obsah adresáře. Rsync klient se připojuje k rsync démonu na stroji hostname (:: značí, že se jedná o spojení na rsyncd), data se budou synchronizovat do uložiště specifikovného názvem modulname (specifikace je věcí konfigurace rsync démona) a adresáře /dest v rámcí modulu.

Vhodný zálohovací rsync příkaz

```
rsync -a --delete-after --rsh='ssh -p55055' /backup buser@bhost.cz:/backup/part1
```

Zálohování se provádí za pomocí vzdáleného shellu ssh, kdy se rsync klient spojí s lokálním ssh klientem a ten naváže spojení se vzdáleným ssh démonem na stroji bhost.cz, na portu 55055, pokusí se příhlásit jako uživatel buser a pokusí se provést synchronizaci adresáře /backup/part1. Tuto metodu zálohy je dobré doplnit ssh autentizaci na bázi klíčů. Všímněte si, že je uvedene **–delete-after**, soubory jsou tedy smazány až na konci relace, pokud něco selže během operací, soubory nebudou odstraněny.

7. Jak si rsyncem značně ušetřit práci

Nyní si ukážeme názorný příklad z praxe, jak může být rsync užitečný a jak může administrátorovi ušetřit mnoho času.
Představte si situaci, že pracujete na postu Linuxového administrátora. Právě dorazila nová farma několika serverových stanic o stejné hardwarové konfiguraci. Před Vámi je teď úkol, co nejrychleji připravit tyto stroje do ostrého provozu. Možností je hned několik, ale já popíšu možnost, kdy pomocí nástroje rsync provedu replikaci jednoho stroje na druhý a ušetřím si tak hromadu práce a monotóní práce.

1. Jeden ze strojů, nazveme jej master, zpracujeme ručně do finálního stavu – install OS, optimalizace OS, install – konfigurace – optimalizace služeb. Dále nastavíme a spustíme rsync démona – vyexportujeme kompletní fs – nezapomenout namountit /boot, postačí pouze pro čtení, budem však potřebovat root privilegia. Provedeme restrikce síťové konektivity na úrovni packet filteringu.
2. Postupně pro každý další slave stroj provedem následující kroky
    1. provedeme boot z liveCD používané distribuce
    2. nakonfigurujeme síť

3. za pomocí ssh provedem dump rozdělení master disků a aplikujem na slave server
   ssh -pX root@master 'sfdisk -d /dev/Y' | sfdisk /dev/Y
4. Vytvoříme správné fs a namountíme slave disky do spravné adresářové struktury, root bude např v /mnt/dst
5. Jedním příkazem rsync provedem kompletní replikaci
   rsync -aH master::exported /mnt/dst
6. Nainstalujem boot loader do MBR
7. Přepíšem konfliktní konfigurace (síť, hostname, …)

Po provedení výše jmenovaných kroku, máte práci na několik hodin hotovou za pár minut, záleží na výpočetním výkonu strojů, vaší zručnosti a především na propustnosti linky.

8. Rsync v akci

Před nějakým časem jsem měl tu možnost řešit zajímavý problém týkající se uložiště dat služby rajce.idnes.cz.

Služba rajce.idnes.cz je postavená na poskytovaní statického obsahu - fotografií, uložiště má následující charakteristiku

- data jednho uživatele jsou uloženy v jednom adreáři
- uložište obsahuje cca 55 000 uživatelů
- uložiště obsahuje celkově cca 1,5TiB dat, průměrná velikost souboru řídově v desítkách KiB
- tedy velké množství adresářů a souborů o malé velikosti, minimálně se měnící
- služba beží na jednom výkonem serveru, možnost pouze horizontální škálovatelnosti

Onehdá docházelo místo na současném uložisti a úkol zněl přemigrovat uživatelská data na nové dynamické uložiště tak, aby to mělo minimální dopad na provoz služby. Služba procházela více změnami a migrace na nížší úrovní by byla velice obtížná.

První nápad zněl rsync, dosavadní i nové uložistě byly lokální, to bude v cuku letu :)

```
nice -n5 rsync -a /stare /nove
```

- pouze seznam souborů má velikost v řádech stovek MiB a ten je duplikovaný na straně přijímače. Migrace musí probíhat za ostrého provozu, zatížení systému je i při použití nice příliš veliké
- nepoužitelné řešení

```
nice -n5 rsync -a  --size-only --whole-file /stare /nove
```

- zjednodušení algoritmu výběru kandidátu na přenos, omezíme se pouze na velikost souboru
- pokud je soubor jiné velikosti, neprovede se rsync algoritmus, ale soubor se pouze celý zkopiruje => ušetřime výpočetní výkon na úkor zatížení IO
- lepší výsledky, ale stále nepoužitelné, zasádní problém v budování seznamu souborů

```
find /stare -type d -maxdepth 1 | while read f; do
      nice -n5 rsync -a  --size-only --whole-file /stare/${f} /nove
done
```

- rsync provádeň po uživateli, minimalizace velikosti seznamu souborů
- použitelné
- ostré migraci předcházelo kopírování po uživateli, beztak ostrá (rsync) migrace trvala více ja 6 hodin

```
find /stare -type d -maxdepth 1 | while read f; do
      nice -n5 cp -a  /stare/${f} /nove
done
```

- nástroj cp s atributem -a (--archive) do značné míry napodobuje chování rsync (rekurzivní, symlinks, práva, vlastníky, časy) ale přenáší všechny data
- mnohem menší systémová režie na úkor IO

9. Závěr

V prácí jsem detailně probral princip fungování nástroje rsync, základu jeho funkčnosti protokol rsync. Specifikoval jsem jasné klady i zápory nástroje, protokolu. Na závěr jsem se věnoval praktickému použití rsync. Já nástroj rsync využívám velice často a pouvažuji jej za jeden z nejmocnějších nástroju pro práci se soubory.

10. Literatura, zdroje

http://samba.anu.edu.au/rsync/
man rsync