Network File System

Michał Wiciński i Emil Jaszczuk

Spis treści

1 Wstęp

1.1 Wstęp i co to

Network File System (NFS) to protokół zezwalający użytkownikom dostęp do plików w sieci tak, jakby były one na lokalnym dysku. Początkowo został on opracowany przez firmę Sun Microsystems (dzisiejsze Oracle) w 1984 roku. Jest to protokół o tak zwanym otwartym standardzie, co oznacza, że każdy może zaimplementować swoją wersję protokołu.

Protokół NFS jest odpowiedzią na brak funkcjonalnego dostępu do serwerowych danych. O ile istniały już wówczas takie usługi jak FTP czy RCP, to szukano dla nich alternatywy, gdzie można by pracować na zdalnych plikach, tak jak na lokalnych.

1.2 Jak działa NFS

NFS działa na zasadzie klient-serwer. Klient po zamontowaniu serwera - prawie tak samo, jakby montował fizyczny dysk - może czytać, tworzyć i edytować pliki. Aby to zrobić, wysyła odpowiednie zapytanie RPC. Zapytanie to wówczas trafia do procesu demona (nfsd) na serwerze i przesyła odpowiedź klientowi. Pakiety są zapisywane w formacie XDR, ponieważ może się zdarzyć, że różne architektury mogą korzystać z innej kolejności bajtów.

1.3 Protokół Stateless

Protokół bezstanowy (stateless) nie posiada informacji o kliencie podczas wykonywania żądań. Z perspektywy serwera, nie wiadomo czy klient aktualnie ma otwarty plik. Mogłoby to się wydawać wadą, jednak ma to swoje zalety, np. jeżeli serwer się zawiesi, nie traci się żadnych potencjalnie wartościowych informacji i można szybko go przywrócić do działania. Dodatkowo, nie musi alokować dodatkowej pamięci na sprawdzanie z jakich zasobów korzystają klienci.

1.4 Protokół Stateful

Protokół stanowy (stateful) natomiast trzyma wszelkie dane o połączonych klientach. Został wprowadzony w wersji NFSv4. Dzięki temu żądania są krótsze, możliwa jest spójność danych w pamięci podręcznej (serwer wie z jakich bloków pamięci korzysta klient).

Obydwie wersje mają swoje plusy i minusy, jednak historia pokazuje, że podejście stateful jest bardziej praktyczne.

2 Historia NFS

Pierwsza wydana wersja NFS była testowana jedynie wewnątrz firmy Sun Microsystems. Tak naprawdę, pierwszym oficjalnym wydaniem tego protokołu był NFSv2.

2.1 NFSv2

NFSv2 został wydany w marcu 1989 roku. Jest protokołem obsługującym żądania synchronicznie, o maksymalnym transferze 8KB na blok. Może korzystać zarówno z TCP jak i UDP. Obsługuje maksymalnie pliki do 2GB, ponieważ jest na 32-bitowej architekturze. Protokół NFS dla wersji drugiej jest już traktowany jako przestarzały.

2.2 NFSv3

Wydany NFSv3 w 1995 roku wprowadził wiele ulepszeń. Między innymi, rozszerzono architekturę z 32-bit na 64-bit. Zwiększono limit transferu, oraz co ważniejsze, wprowadzono asynchroniczność. Dodatkowo, poprawiono obsługę błędów. NFSv3 jest dalej najczęściej używaną wersją NFS, mimo że wyszły nowsze.

2.3 NFSv4

NFSv4 wydany w 2003 roku nie był już rozwijany przez Sun Microsystems, a przez IETF (Internet Engineering Task Force). Był to stosunkowo duży przełom dla protokołu. Po pierwsze, wprowadzono stateful. Odtąd, korzystano tylko z jednego portu, gdzie we wcześniejszych wersjach dla np. protokołu Mount czy Status numer był przydzielany dynamicznie. Dodatkowo, zintegrowano blokowanie plików do samego protokołu, a nie tak jak w NFSv3 korzystano z NLM (Network Lock Manager). Z uwagi na wiele zmian w tej wersji i potencjalne problemy z modyfikacją konfiguracji, wielu administratorów nie zdecydowało się na aktualizację z NFSv3.

3 Bezpieczeństwo

3.1 Wstęp

Zakładając, że nasza sieć nie ma dostępu do świata zewnętrznego, to pliki na serwerze NFS są relatywnie bezpieczne. Jednak naturalnie szansa na taką sytuację, zwłaszcza w dzisiejszych czasach, jest praktycznie zerowa.

W NFS (zwłaszcza w wersji drugiej i trzeciej) możemy napotkać na różne problemy związane z bezpieczeństwem. Przykładowo, próbując zamontować dysk na kliencie, wysyłamy zapytanie do serwera. Ten zaś przechowuje dane o kliencie, między innymi jego adres IP. Nie jest to najbezpieczniejsze rozwiązanie na świecie, ponieważ atakujący może się łatwo podszyć pod klienta (spoofing).

Kolejny scenariusz: Użytkownik A o UserID 9999 tworzy plik, do którego tylko on ma dostęp. Użytkownik B może zmapować swój UserID na 9999, przez co serwer będzie myślał, że klient B jest tak naprawdę klientem A.

NFS sam w sobie oferuje raczej niewiele rozwiązań w celu poprawy bezpieczeństwa. Jest jednak kilka stosunkowo prostych metod rozwiązujących ten problem.

3.2 Portmapper

Ponieważ NFS używa protokołu RPC, usługa portmappera musi być włączona (przynajmniej dla wersji NFSv2 i NFSv3, wersja czwarta nasłuchuje na ogólnie znanym porcie 2049). Portmapper umożliwia komunikację klientów z serwerem i śledzi porty usług RPC. Dzięki plikom `/etc/hosts.allow` oraz `/etc/hosts.deny` możemy kontrolować ruch i uniemożliwić dostęp do serwera niechcianym maszynom. Przykładowo, możemy otworzyć usługę tylko maszynom w danej podsieci.

Zamiast hosts.allow / hosts.deny nalezy uzyc firewalla (np. IPTabes)https://fed

3.3 Nsfd i mountd

Po stronie serwera powinniśmy mieć domyślnie włączoną usługę `root_squash` w pliku `/etc/exports`. Jeżeli użytkownik ma UID 0, serwer zmapuje jego UID na anonimowy, dzięki czemu nie będzie mógł czytać lub edytować plików dostępnych dla roota po stronie serwera.

3.4 Firewall

IPTables / nftablesM

Warto również mieć włączone usługi IPchains (firewall) oraz netfilter (monitorowanie i zmienianie ruchu sieci). Zapewniają one bezpieczeństwo niższego poziomu, w porównaniu z domyślnym demonem NFS, dzięki czemu można szybciej powstrzymać potencjalny atak.

4 Inne sieciowe systemy plików

Poza NFS istnieje również wiele innych podobnych technologii. Naturalnie, każdy z nich ma wady i zalety. Umiejętnością administratora jest wybranie systemu odpowiadającego wymaganiom.

4.1 SMB

SMB (Server Message Block) to podobny protokół do NFS. Znany jest też jako CIFS (Common Internet File System). Również jest protokołem typu klient-serwer. O ile benchmarki wykazują zbliżone prędkości przesyłu plików, SMB może być lepszym wyborem dla ludzi korzystających z systemu Windows, jako że wsparcie jest o wiele lepsze.

4.2 Lustre

Lustre to rozproszony system plików (ang. Distributed File System, DFS), głównie używany przy klastrach komputerowych. Oznacza to, że pliki mogą być rozrzucone po wielu różnych serwerach, lecz dla użytkownika (klienta) będą widoczne jako jedna, spójna całość. Swoją popularność zawdzięcza dzięki stosowaniu tego systemu przez superkomputery takie jak Fugaku czy Titan. W przeciwieństwie do NFS, bardzo dobrze się skaluje. Na pewno będzie lepszym wyborem do zastosowań komercyjnych, gdzie przetwarzane są ogromne ilości danych. Oferuje również bardzo dużą prędkość odczytu i zapisu. Jego największym problemem jest trudna konfiguracja i utrzymanie.

4.3 GlusterFS

Gluster to kolejny rozproszony system plików, który może korzystać z wielu różnych serwerów. Podobnie jak Lustre, bardzo dobrze się skaluje do nawet kilku petabajtów danych oraz ma wiele dodatkowych zabezpieczeń, których NFS (jako sam protokół) nie oferuje. Przez to jednak - według benchmarków - odczyt i zapis plików jest wolniejszy.

5 Konfiguracja NFS

5.1 Po stronie serwera

Najpierw instalujemy narzędzia potrzebne do działania serwera NFS.

```
[michal@localhost /]$ sudo yum -y install nfs-utils
[sudo] hasło użytkownika michal:
Ostatnio sprawdzono ważność metadanych: 1:01:46 temu w dniu sob, 7 lis 2020,
14:48:04.
Pakiet nfs-utils-1:2.5.1-4.rc4.fc31.x86_64 jest już zainstalowany.
Rozwiązano zależności.
Nie ma nic do zrobienia.
Ukończono.
```

Następnie uruchamiamy i włączamy zainstalowaną usługę.

```
[michal@localhost /]$ sudo systemctl start nfs-server.service
[michal@localhost /]$ sudo systemctl enable nfs-server.service
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service
-> /usr/lib/systemd/system/nfs-server.service.
```

Tworzymy folder, w którym będziemy udostępniać pliki i ustawiamy do niego uprawnienia, żeby nie było problemu z dostępem.

```
[michal@localhost /]$ sudo mkdir -p /srv/nfs/pub
[michal@localhost /]$ sudo chown -R nobody: /srv/nfs/pub
[michal@localhost /]$ sudo chmod -R 777 /srv/nfs/pub
```

Zapewniamy klientom dostęp do serwera poprzez stworzenie pliku `/etc/exports` wpisując następujące dane:

```
[michal@localhost etc]$ sudo nano /etc/exports
[sudo] hasło użytkownika michal:
[michal@localhost etc]$ cat /etc/exports
/srv/nfs/pub 192.168.134.225(rw,sync,no_all_squash,root_squash)
```

Adres IP różni się w zależności od klienta. Można dodać ich wiele.

`rw` - pozwala na odczyt i zapis współdzielonych plików
`sync` - zapisuje każdą zmianę na dysku przed jej zastosowaniem
`no_all_squash` - mapuje wszystkie identyfikatory UID i GID z żądania klienta na identyczne identyfikatory UID i GID na serwerze NFS
`root_squash` - mapuje żądania od użytkownika root po stronie klienta na anonimowy UID/GID

Następnie eksportujemy współdzielone pliki do klienta.

```
[michal@localhost etc]$ sudo exportfs -rav
exporting 192.168.134.225:/srv/nfs/pub
```

Oraz zezwalamy na działanie usług potrzebnych do NFS w zaporze ogniowej serwera.

```
[michal@localhost etc]$ sudo firewall-cmd --permanent --add-service=nfs success
[michal@localhost etc]$ sudo firewall-cmd --permanent --add-service=rpc-bind
success
[michal@localhost etc]$ sudo firewall-cmd --permanent --add-service=mountd
success
[michal@localhost etc]$ sudo firewall-cmd --reload success
```

Logi na serwerze są przechowywane w pliku `/var/log/messages`.

5.2 Po stronie klienta

Instalujemy narzędzia potrzebne do działania serwera NFS.

```
[maciej@localhost ~]$ su
Password:
[root@localhost maciej]# yum -y install nfs-utils
Last metadata expiration check: 2:01:34 ago on Sat 07 Nov 2020 02:21:14 PM CET.
Package nfs-utils-1:2.5.1-4.rc4.fc31.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
```

Tworzymy folder, w którym będzie udostępniona zawartość serwera. Możemy to zrobić, gdzie chcemy.

```
[root@localhost maciej]# mkdir -p /srv/nfs/shared_folder
```

Następnie montujemy udostępniony katalog u klienta. Adres IP oczywiście zależy od serwera.

```
[root@localhost srv]# mount 192.168.134.215:/srv/nfs/pub /srv/nfs/shared_folder
```

5.3 Weryfikacja rozwiązania

Tworzymy plik tekstowy po stronie klienta.

```
[maciej@localhost shared_folder]$ cat > sprawdzam.txt
```

```
test
^C
[maciej@localhost shared_folder]$ cat sprawdzam.txt
test
[maciej@localhost shared_folder]$ ls -l
total 4
-rw-rw-r--. 1 maciej maciej 5 Nov 7 16:39 sprawdzam.txt
```

Sprawdzamy, czy plik pojawił się na serwerze.

```
[michal@localhost pub]$ ls -l
razem 4
-rw-rw-r--. 1 michal michal 5 11-07 16:39 sprawdzam.txt
[michal@localhost pub]$ cat sprawdzam.txt
test
```

Właścicielem pliku jest użytkownik na serwerze.

6 Bibliografia

[1] www.extrahop.com/resources/protocols/nfs/
[2] www.cs.rutgers.edu/~pxk/416/notes/18-nfs.html
[3] searchenterprisedesktop.techtarget.com/definition/Network-File-System
[4] www.quora.com/How-does-NFS-Network-File-System-work
[5] euclid.nmu.edu/~rappleto/Classes/CS442/Notes/nfs.html
[6] https://www.quora.com/How-does-NFS-Network-File-System-work
[7] tldp.org/HOWTO/NFS-HOWTO/security.html
[8] ferhatakgun.com/network-share-performance-differences-between-nfs-smb/
[9] wiki.lustre.org/NFS_vs._Lustre
[10] docs.gluster.org/en/latest/Administrator%20Guide/GlusterFS%20Introduction/
[11] serverfault.com/questions/372151/nas-performance-nfs-vs-samba-vs-glusterfs
[12] vitux.com/install-nfs-server-and-client-on-ubuntu/
[13] www.redhat.com/sysadmin/nfs-server-client