分散システム特論レポート

Network File System Protocol Specification

[RFC1094],[RFC1813],[draft-ietf-nfsv4-07]

伊藤 政利 (masatosi@swlab.csce.kyushu-u.ac.jp)

提出日 2001 年 2 月 28 日

1 概要

NFS はクライアントアプリケーションからネットワーク上のサーバのファイルに、透過的なファイルアクセスを可能にするプロトコルである。計算機や OS、ネットワークアーキテクチャや転送プロトコルには様々なものが存在するが、NFS はそれらに依存すること無く動作する。なお、NFS プロトコルは RPC(Remote Procedure Call) や XDR(eXternal Data Representation)、そしてマウントプロトコルを用いる。

2 RPC(Remote Procedure Call) [RFC1831]

RPC とはクライアント・サーバモデルのネットワーク環境において、システム間で通信を行うためのプロトコルである。この RPC を使用したネットワークプログラミングにおいては、クライアントプログラムは、リモートホスト上の関数 (サブルーチン) をネットワーク経由で呼び出すように記述される。

また、RPC サーバプログラムはウェルノウンポートではなく、エフェメラルポート (サービスを受けている間だけ存在するポート) を利用する。その際、どの RPC プログラムがどのエフェメラルポートを利用しているかを監視するものが必要であり、これをポートマッパーと呼ぶ。”rpcinfo” というコマンドを使うことで、現在のマッピング状況を調べることができる。

2.1 RPC の利点

RPC を用いることにより、以下のような利点がある。

- ネットワークプログラミングのほとんどの部分が省けるので、プログラミングが容易になる。アプリケーションプログラマはクライアントプログラムと、クライアントが呼び出すサーバプロシージャだけを記述すれば良い。
- 信頼性の必要な通信において、UDP などの信頼性のないプロトコルを使うとき、再送やタイムアウトなどが必要となるが、それらは全て RPC によって行われる。
- RPC ライブラリは引数と帰り値に要求される、あらゆるデータ変換を取り扱う。

3 XDR(eXternal Data Representation) [RFC1832]

XDR とは RPC コールと応答メッセージの値をコード化するための標準である。それらの値をコード化する標準方法を定義することで、あるシステム上のクライアントからアーキテクチャの異なる他のシステム上のプロシージャを呼び出すことが可能になる。

4 NFS(Network File System)

NFS はクライアントに、サーバ上のファイルに対する透過的なファイルアクセス機能を提供する。これにより、ローカルファイルを利用するクライアントアプリケーションが、プログラムファイルに何の変更を加えることなく、NFS ファイルも利用することができる。

4.1 NFS マウント

クライアントは、サーバのファイルシステムのファイルにアクセスするため、サーバのファイルシステムをマウントしなければならない。それにより、クライアントはサーバのファイルシステムに対するファイルハンドルを入手することができる。

NFS マウントの手順は図 1 のようになる。

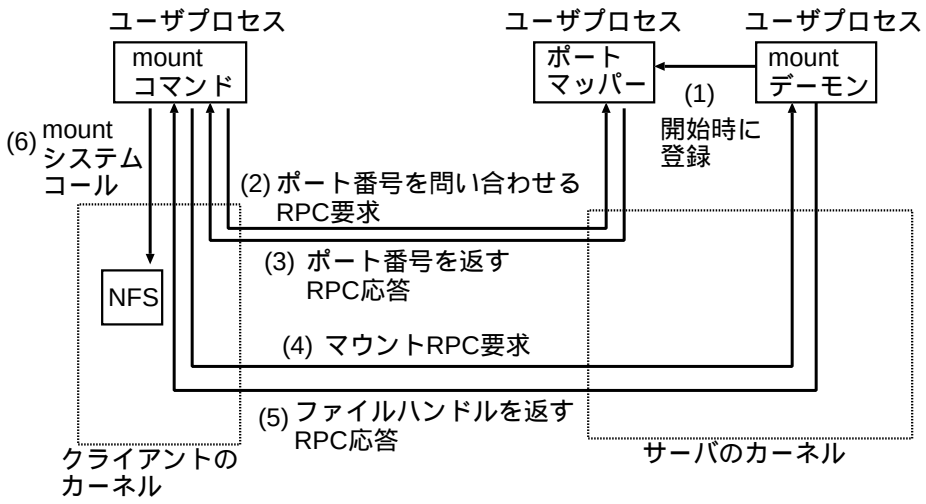

図 1: NFS マウントの手順

1. マウントデーモン (mountd) が起動し、ポートマッパーにポート番号を登録する。
2. クライアント上で mount コマンドが実行されると、ポートマッパーに対して RPC コールを発行し、サーバのマウントデーモンのポート番号を入手する。
3. ポートマッパーがポート番号を返す。
4. mount コマンドは、マウントデーモンに対してサーバ上のファイルシステムにマウントするための RPC コールを発行する。サーバはクライアントの IP アドレス・ポート番号をチェックして、指定したファイルシステムへのマウントを許可するかどうかを判断する。

5. マウントデーモンは、該当のファイルシステムのファイルハンドルを返す。
6. ステップ 5 で返されたファイルハンドルとクライアント上のローカルマウントポイントを連携させるために、mount コマンドはクライアント上で mount システムコールを発行する。サーバのファイルシステム上のファイルに対するユーザプロセスの参照は、そのファイルハンドルが開始点として利用されるようになる。

以上の流れにより、NFS マウントを行う。NFS マウントの例として自分が普段使用している環境をあげる。自分の場合、図 2 のように NFS マウントを行っている。

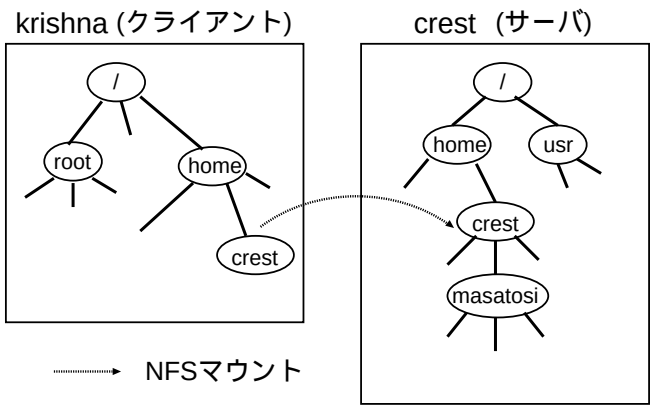

図 2: NFS マウントの例

また、NFS の仕組みを簡単に表すと図 3 のようになっている。

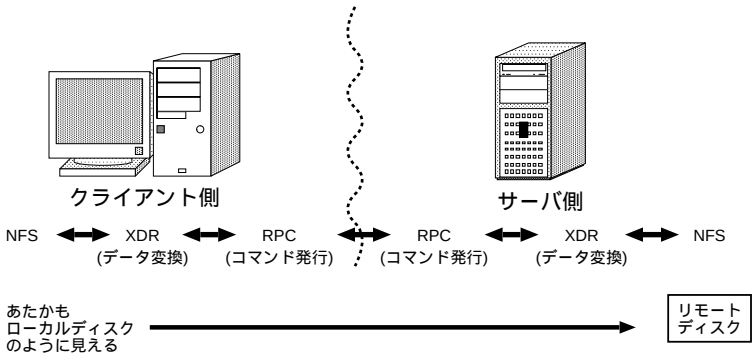

図 3: NFS の仕組み

・NFS のプロトコル体系

NFS	クライアントはローカルファイルにも NFS ファイルにも透過的にアクセスできる。
XDR	アーキテクチャの違いによって異なるデータ形式を標準形式に変更し、ローカルコンピュータ側で再変換する。
RPC	サーバに要求を出し、何らかのプロシージャを実行させ、結果を受け取る。

4.2 プロシージャ

NFS は多くのプロシージャを提供している。

- Version2[RFC1094] では 18 個
- Version3[RFC1813] では 22 個
- Version4[draft-ietf-nfsv4-07] では 37 個

Version4 では実際は NULL と COMPOUND という 2 つのプロシージャのみであるが、COMPOUND というプロシージャの中に、Version2 や Version3 で言うところのプロシージャを「オペレーション」として 36 個含んでいる。

プロシージャの例

以下にプロシージャの例とその簡単な意味をあげる。

- NULL
  - 何もしない
  - テストなどに用いられる
- ACCESS
  - ファイルアクセス許可のチェック
  - Version3 からの新しいプロシージャ
- LOOKUP
  - ファイルの探索
- MKDIR
  - ディレクトリの作成
- CREATE
  - ファイルの作成
- READ
  - ファイルの読み込み
- WRITE
  - ファイルの書き込み

他にも多数のプロシージャが存在するが、ここでは後に出てくるものを主に紹介した。

4.3 Version2 と Version3 との違い

NFS が Version2 から Version3 になった際の変更点について述べる。

- ファイルハンドルが 32 バイトの固定長から最高 64 バイトの可変長になった。
- READ や WRITE のプロシージャにおいて、データ転送できる最大の大きさが 8192 バイトから無制限になった。
- READ および WRITE プロシージャのファイルサイズと先頭バイトオフセットが 32 ビットから 64 ビットになった。
- WRITE で同期させる必要がなくなった。

他にも多数の変更点があるが、一部を抜粋した。

4.4 Version4 と以前の Version との違い

NFS の Version4 と それ以前の Version (Version2, Version3) との違いについて述べる。

- セキュリティ
  - 端末計算機間のセキュリティのため、基本的な RPC のセキュリティを拡張する RPCSEC_GSS framework [RFC2203] を使用する。
  - Kerberos V5 [RFC1964] を使用する。
- オペレーション
  - プロシージャは NULL と COMPOUND のみになる。
  - 今までプロシージャと呼ばれていたものはオペレーションになる。
  - COMPOUND プロシージャの中に複数のオペレーションを結合させる。これにより、クライアントサーバ間のやり取りを減少させることができる。
- ファイルハンドル型
  - 以前のバージョンでは、ファイルハンドルの存続時間が長い間有効であり、サーバの実装によっては、この持続性に関する要求に合わせるのが難しいサーバもあった。そこで、今までの持続性のあるファイルハンドルと揮発性 (volatile) のファイルハンドルという 2 つの型を導入し、環境によって合わせることができる。

他にも非常に多くの変更点があるのが、ここでは一部を抜粋した。

5 実装例

NFS は多くのシステムに実装されており、実装例としては以下のようなものがある。

- FreeBSD のカーネル内に実装
- SunOS のカーネル内に実装

- Linux のカーネル内に実装

図 2 において、クライアントの krishna は FreeBSD 4.0-Release で、サーバの crest は Solaris 2.6 (SunOS 5.6) である。また、使用している NFS のバージョンは 3 である。

この環境の下、tcpdump を実行させてある状態で、krishna において以下のコマンドを実行してみる。

```
% mkdir /home/crest/masatosi/tmp
```

%はコマンドプロンプトを表し、tmp は新たなディレクトリである。この時の tcpdump の出力の一部を以下に示す。その際、説明に必要な箇所だけを抜粋した。

1. krishna.273617024 〉 crest.nfs: 116 access fh 1299617
2. crest.nfs 〉 krishna.273617024: reply ok 120 access
3. krishna.273617025 〉 crest.nfs: 124 lookup ”masatosi”
4. crest.nfs 〉 krishna.273617025: reply ok 240 lookup fh 1292805
5. krishna.273617026 〉 crest.nfs: 120 lookup fh 1292805 ”tmp”
6. crest.nfs 〉 krishna.273617026: reply ok 116 lookup ERROR: No such file or directory
7. krishna.273617027 〉 crest.nfs: 148 mkdir ”tmp” fh 1292805
8. crest.nfs 〉 krishna.273617027: reply ok 272 mkdir fh 461088

上記の出力における各行は以下のような意味を表す。

1. クライアントの krishna からサーバの crest に ACCESS を行う。273617024 はトランザクション ID を表し、1299617 はファイルハンドルを表す。
2. crest から krishna に OK と返す。
3. krishna から crest に”masatosi”を LOOKUP で検索する。
4. crest から krishna に OK を返し、”masatosi”に対するファイルハンドルを返す。
5. krishna から crest に、ファイルハンドル 1292805 を使い、”tmp”を LOOKUP で検索する。
6. crest から krishna に ERROR: No such file or directory と返す。
7. krishna から crest に MKDIR で”tmp”のディレクトリを作成する。
8. crest から krishna に MKDIR に対する OK を返す。