高信頼ネットワークファイルシステムの設計および実装*

6C－5

榎戸 智也　酒寄 彰彦　田島 真　桧垣 博章　滝沢 誠†

東京電機大学‡

e-mail{eno, aki, mako, hig, taki}@takilab.k.dendai.ac.jp

1　はじめに

グループウェアなどの分散型の応用では、ネットワーク上にあるファイルの共有を行なう必要がある。ファイルが複数の計算機に分散したシステムでは、利用者は各ファイルの所在、アクセス方法と独立にファイルをアクセスできることが必要である。即ち、利用者は、ファイルシステムをあたかも一つの計算機内にあるかのようにアクセスできる必要がある。本論文では、全ての計算機に対して、一つの仮想ファイルシステムを提供することを考える。また、サーバの障害に対して、ファイルサービスを提供し続けるネットワークファイルシステムについて検討する。

2　システムモデル

2.1　仮想ファイルシステム

本システムは、ネットワーク全体で1つの仮想ファイルシステム（*VFS*）を提供する。利用者は、ファイル操作用の利用者インタフェース *R_shell* を用いて *VFS* にアクセスを行う。*VFS* 内の部分木 V_i を単位として、サーバに記憶される。V_i が、サーバ S_1, ..., S_m ($m \geq 1$) 内に記憶されるとする。S_j に記憶された部分木 V_i を V_{ij} とする。V_{ij} を V_i のレプリカとする。レプリカ V_{i1}, ..., V_{im} の集合をクラスタとする。部分木 V_i は、V_i の根のノードのディレクトリ名により示す。クラスタ $\{V_{i1}, ..., V_{im}\}$ の名前を V_i とする。部分木間の関係は、レプリカ間のポインタにより示される。$V_i = \{V_{i1}, ..., V_{im_i}\}$ と $V_j = \{V_{j1}, ..., V_{jm_j}\}$ を考える。V_j は V_i の部分木とする。この関係は、V_{ih} から V_{jk} へのポインタ ($h = 1, ..., m_i$, $k = 1, ..., m_j$) として示される。

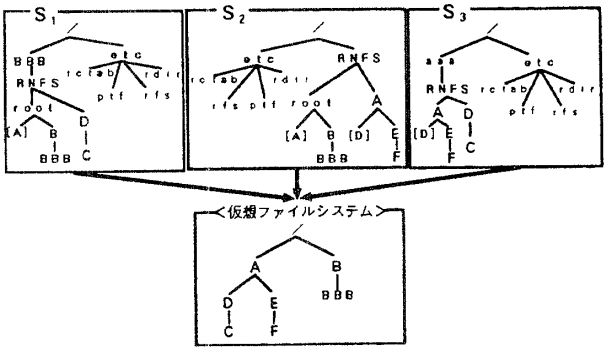

図 1: ファイルシステム

図1に、*VFS* の構成例を示す。各サーバは、それぞれのファイルシステムの任意の位置に、*VFS* の部分木を記憶するためのディレクトリ *RNFS* を持つ。サーバ S_1 は、*VFS* のルート (/)、とディレクトリ *B* と *D*、ファイル *BBB* と *C* を記憶している。/ の下の [*A*] は、ディレクトリ *A* へのポインタを示す。*A* はサーバ S_2 と S_3 に存在し、[*A*] は、これらへのポインタである。/ は、S_1 と S_2 にレプリカがある。S_2 の / の下の [*A*] は、S_2 内の *A* と S_3 内の *A* を示す。

2.2　システムの構成

システムは、*R_shell*、*R_process*、UNIX カーネルから構成される。*R_shell* は、利用者が *VFS* をアクセスするためのインタフェースである。*R_shell* は、利用者が *VFS* にアクセスするために発行したコマンド (*R_*コマンド) を、アクセス対象となるクラスタ内のサーバに送信する。さらに、*R_shell* は、*VFS* に対して行なったコマンドの実行結果を利用者に返す。*R_process* は、他および自分のクラスタ内のサーバ間で通信を行なう。利用者がディレクトリの移動やファイル操作を行なう際、レプリカを持つサーバ間で通信を行なう必要がある。さらに、同一クラスタ内のレプリカで行なわれる処理の同期をとるための通信を行なう。*R_process* は、上述の通信機能の他に *R_shell* から受信したコマンドおよびパス名等を解析し、所有するレプリカ内のファイル操作を行なう。

2.3　ポインタ管理

図1に示した *VFS* を考える。各サーバでの *RNFS* までのパス名 */…/RNFS* は、ファイル */etc/rdir* に記述される (S_1:*/BBB/RNFS*, S_2:*/RNFS*, S_3:*/aaa/RNFS*)。各レプリカ V_i へのポインタは、V_i が存在する計算機名 S_i と *RNFS* 以下のディレクトリ名 */… /RNFS/… /*d_i の組である。ここで、d_i は V_i の内のローカル名である。各サーバは、*VFS* のルート / を示す *root* レプリカへのポインタを持ち、これらファイル */etc/rfs* に記述される (ポインタ元 = *VFS* の /、ポインタ先 = S_1, S_2 : root)。

他のサーバ内のレプリカへのポインタ、ポインタファイル */etc/ptf* に記述する。子から親レプリカへのポインタは、クライアントが保持する。各サーバが持つファイル *rdir*、*rfs*、*ptf* と、クライアントが管理する子クラスタから親クラスタへのポインタ情報によりクラスタ間のレプリカの関連を実現できる。クラスタレプリカの一貫性を保つため、レプリカ間で通信を行なう必要がある。このために、同一クラスタ内のレプリカ間のポインタが必要となる。各サーバのファイル */etc/rctab* にレプリカ間のポインタが記憶される。

3　アクセス方法

3.1　ディレクトリの移動

VFS 内でのディレクトリの移動は、サーバ内のレプリカ内、またはレプリカ間の移動として実現される。このために、クライアントは、以下の情報を管理する。

- *VFS* 内のカレントワーキングディレクトリ (*cwd*)。
- *cwd* が存在するカレントワーキングクラスタ (*cwc*)。
- *cwc* の示すレプリカ内の *cwd* のパス名 (*cwdc*)。
- カレントレプリカ (*crep*)。

---

*Reliable Network File System using Inter-Cluster Communication

†Tomoya Enokido, Akihiko Sakayori, Makoto Tajima, Hiroaki, Higaki, and Makoto Takizawa

‡Tokyo Denki University

cwd に、*VFS* 内のディレクトリを示す。*VFS* 内の移動は、*cd* コマンドによって行なわれ、移動により *cwd* が変化する。*cwc* は、*cwd* に示すディレクトリがどの部分木に存在するかを示している。*cwd* の示す部分木が、レプリカとして、サーバに配置されている。図1で、ルートで '*cd A/D*' コマンドを利用者が実行したとする。この結果、$cwd = /A/D$、$cwdc = /D$、$cwc = cwdc$ となる。*cd* コマンドに対して、クラスタ内のある1つのレプリカをカレントレプリカ (*crep*) とする。

3.2 ファイルアクセス方法

ファイル F に対する操作を考える。クラスタ C の主サーバは、ファイルの操作に対して、レプリカ $F_1, ..., F_m$ の一致性を保つための制御を行なう。主サーバ S_1 は、ファイル操作要求に対して、レプリカ $F_1, ..., F_m$ をロックする。この後に、レプリカの操作を行なう。まず、前節で述べた手順により、操作対象のレプリカ F_i が特定される。読み込み、更新、作成、削除の方法を示す。

A. 読み込み

対象レプリカ F_i を R モードでロックする (*Rlock*)。ロックできたならば、クライアントは F_i を読み出す。

B. 更新

ファイル F の更新について考える。対象レプリカ F_i がクライアントにまず読み出される。クライアントで変更が行なわれる。ここで、全レプリカ $F_1, ..., F_m$ が W モードでロックされる (*Wlock*)。ロックされたならば、更新結果が、全レプリカ $F_1, ..., F_m$ に送られ、更新される。このために、クライアントは更新が行われたファイルを主サーバ S_1 にファイル *F.new* として、コピーする。さらに、S_1 は、更新が行なわれたファイルを各サーバ S_i にコピーする。ファイルのコピーを行った S_i は、主サーバ S_1 に正常終了を通知する。S_1 は、全サーバ $S_2, ..., S_m$ から正常終了通知を受け取った後、全サーバに対して、ファイル *F.new* を F に改名する通知をする。

この通知を受けた各サーバは、更新後のファイル名を更新前のファイル名にし、更新前のファイルを削除する。ファイルの改名処理が正常終了したサーバは、主サーバ S_1 に改名処理の正常終了を通知する。S_1 は、クラスタ内の全サーバから改名処理が正常終了したことを受信した後に、クライアントに対して更新が正常に行われたことを通知する。主サーバが、ファイルのコピーまたは改名処理において、全サーバから正常終了の通知を受信出来なかった場合は、クライアントに更新が失敗したというメッセージを送信する。図2に、クラスタ A 内に存在するファイルを更新する際の手順を示す。

C. ファイルの作成

新しいファイル Fは、クライアント C内に作成される。ディレクトリは、*mkdir*コマンドで作成される。ファイルおよびディレクトリの作成要求を受けたクライアント Cは、クラスタ内の主サーバ S_1 に対して、メッセージを送信する。B で述べたと同様の手順で、ファイル、ディレクトリを作成する。S_1 は、ファイル、またわディレクトリ *F.new*を作成し、$S_1, \cdots S_m$ に *F.new*を作成させる。全てのサーバで作成できたならば、*F.new*を Fとする。

D. ファイルの削除

利用者は、*rm*、*rmdir*等のコマンドを用いてファイル Fの削除を行なう。クライアント Cは、対象クラスタ内の主サーバ S_1 に対して、削除要求メッセージを送信する。このメッセージは、コマンドおよび特定されたクラスタ内で削除を行うファイルまでのパス名を含む。

1 S_i は、Fを *F.bak*と改名する。S_i は、完了通知を送信する。

2 S_1 が完了通知を全サーバから受信したならば、S_i に削除要求を行う。

3 S_i は、*F.bak*を削除する。

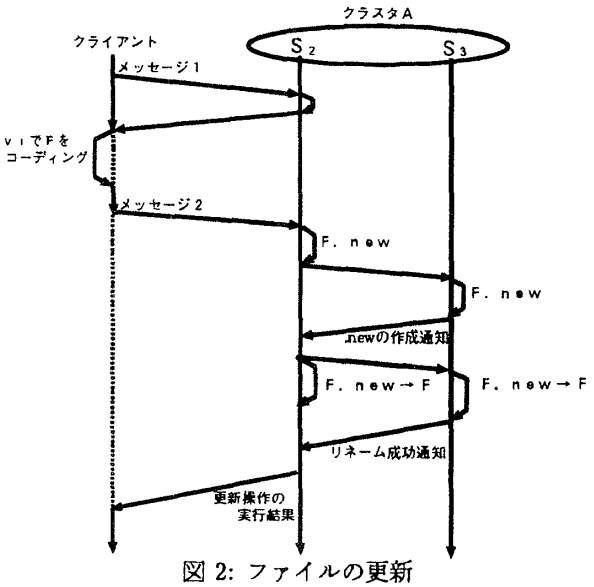

図 2: ファイルの更新

4 評価

ここでは、更新処理の性能について評価を行なう。全てのファイルレプリカの更新処理が終了するまでの時間を更新時間とする。対象ファイルレプリカ数を n とする。図 3に、対象ファイルレプリカ数 n に対する更新時間 t を示す。図から、*RNFS* におけるファイルの更新時間は、対象ファイルレプリカ数に比例することがわかる。また、*RNFS* におけるファイルの更新時間は、更新対象ファイルサイズに比例する。

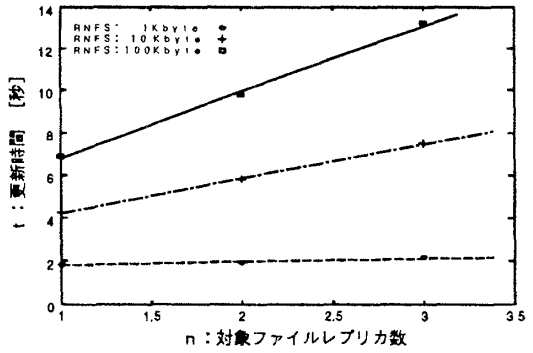

図 3: ファイルレプリカ数に対する更新時間

5 まとめ

本論文では、システム内のある計算機に障害が発生しても、利用者には障害を意識させない高信頼ネットワークファイルシステム *RNFS* の設計と実装について論じた。信頼性を提供するための手法として、各計算機内のファイルシステムを複数のサーバに多重化する方法を述べた。

参考文献

[1] Shima, K., Higaki, H., and Takizawa, M., "Fault-Tolerant Intra Group Communication," *Proc. of the 10th IEEE ICPADS*, 1996, pp. 467–473.

[2] "NFS: Network File System Protocol Specification(RFC 1094)," *SunMicrosystems Inc*, 1989, pp. 1–26.