2S-5

アクセス頻度によってデータを再配置するサーバレス分散ファイルシステム

藤井孝文　安倍広多　石橋勇人　松浦敏雄
大阪市立大学 大学院創造都市研究科

1. はじめに

大学や企業等の組織ではファイルサーバを設置してユーザのデータを格納する場合が多いが、ファイルサーバは負荷が集中しやすく、また、物理的、ネットワーク的に単一故障点にもなる。さらに、単体のファイルサーバで多数のクライアントに対応できるサーバは非常に高価である。

一方、最近の PC は処理能力も向上しており、比較的大容量のディスクを有している。また、上記のような組織では PC が多く存在している。そこで、PC のディスクの空き領域を組み合わせ仮想的なファイルサーバを構築できれば、単一のファイルサーバと置き換えることができ、上記の問題を解決できると考えられる。

本研究の目的は上記のように PC のディスクを組み合わせた仮想ファイルサーバのファイルシステム VDFS を提案しその有効性を確認することである。本稿では VDFS の設計と実装について述べる。

2. 設計

2.1 設計方針

VDFS は PC を組み合わせたファイルシステムである。VDFS は組織内での利用に適したファイルシステムを目指す。VDFS を構成している PC(以下ノードと呼ぶ)は管理者が管理しており、ユーザにはノードの管理権限がないことを前提とする。これらノードは高速な LAN で接続されているとする。また、ノードの数は一定ではなく、故障や新規導入等により増減する。

分散ファイルシステムは様々なものが研究されている。広域ネットワークでの利用を想定しているもの([1]等)をそのまま LAN 上で利用するとオーバヘッドが大きいと考えられる。また、LAN 上での利用を想定しているものもあるが([2]等)、ノードの増減が困難である。

VDFS は前に述べた前提の下で以下のような方針を立て設計を行った。

方針 1. 単一故障点になるような管理ノードは置かない。
方針 2. ノードは任意の時点で増減できる。
方針 3. ノードの故障等を考慮し、冗長性を確保するためにファイルの複製を作成し、複数のノードに配置する。
方針 4. ファイルのアクセス効率を高めるため、ファイルのアクセス頻度によって複製の配置ノードを変更する。

2.2 VDFS の概要

図 1 が VDFS のイメージ図である。ユーザはノードの一台を利用し、VDFS の仮想的なボリュームをマウントしてファイルにアクセスする。例えば VDFS にファイル a, b が存在する場合を考える。図では a は 3 つのブロックに、b は 2 つのブロックに分割され、各ブロックは複製が作られて複数のノードに配置される。図では a のブロック#1 はノード A, D に配置されている。また、どのブロックがどのノードに配置されているかという情報(位置情報)も同様に複製が作られ、複数のノードに配置される。ファイルを利用する時はこの情報を元にブロックを特定する。

以下、VDFS のノードの管理やファイルの扱い等について述べる。

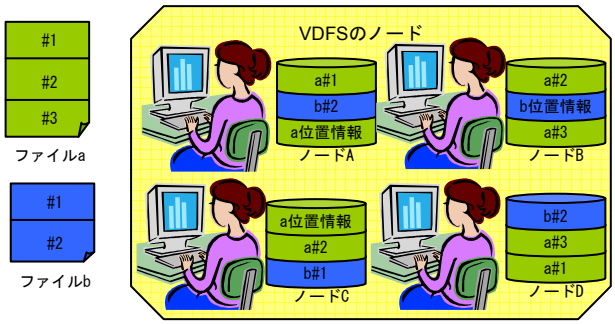

図 1. VDFS イメージ図

2.3 ノード管理

VDFS ではノードを自由に追加・削除できるため、現在 VDFS を構成しているノードを常に把握しておく必要がある。また、全てのノードが動作中とは限らないため、効率よくブロックを取得するために各ノードの動作状況も把握しておくことが望ま

A Serverless Distributed Filesystem Relocating Data by Access Frequency
Fujii Takafumi, Abe Kota, Ishibashi Hayato, Matsuura Toshio
Graduate School for Creative Cities, Osaka City University

しい。このため、VDFS ではノードリスト(各ノードの情報を記録したもの)を各ノードに巡回させることで、全ノードに情報を知らせる。全ノードがノード集合の情報を持つことで毎回ノード状態をチェックする必要をなくす。ノードリストの巡回は定期的に行う。また、定期的な巡回以外にも他のノードの停止を検知したノードは自身のノードリストを更新し、巡回を開始する。各ノードの識別にはノード ID を利用する。ノード ID は各ノードが独立して生成しても衝突しないように、IP アドレスやホスト名等を組み合わせたものをハッシュした値を用いる。

2.4 ファイルの表現方法

VDFS では、ファイルを固定長のブロック(データブロック)に分割してノードに配置する。また、構成データと呼ばれる、データ識別情報(後述)やファイルの所有者情報等の情報を記録したデータも配置する(図 2)。データ識別情報はデータ ID(個々のデータブロック及び構成データを識別するためのもの)とそのデータを配置したノードのノード ID(これは複製の数だけ書かれる)からなる(図 2(1))。ディレクトリエントリにはデータ識別情報とファイル名が記録され、ファイルの配置ノードが特定できる。図 2(2)では abc/xyz ファイルはノード ID6321,3762 にあることがわかる。データ ID は、各ノードが独立に作成しても衝突しないように、ノード ID や時刻等から作成する。

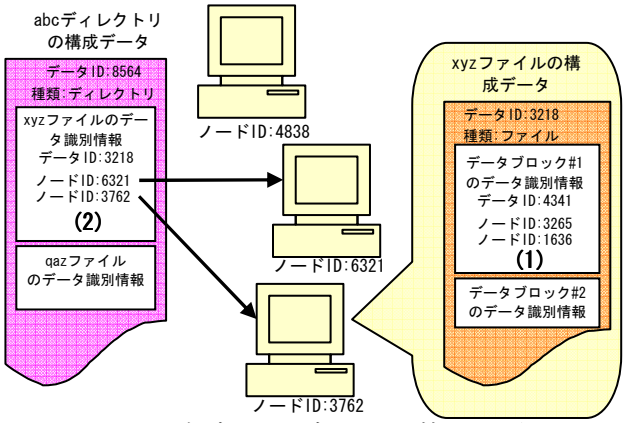

図 2. 構成データとデータ識別情報の関係

2.5 データアクセス

ファイルにアクセスする場合は、まず、ファイルのパス名からファイルのデータ識別情報を特定する。パス名の検索はルートディレクトリから始めることになるが、ルートディレクトリのデータ識別情報はノードリストに含まれておりノードリスト巡回時に取得する。また、一度特定したファイルのパス名とデータ識別情報の組み合わせはノードの名前キャッシュに保存し再利用する。

ファイルのデータ識別情報を特定した後に、対象のデータが自ノードにある場合はそれを利用し、そうでない場合はデータ識別情報に載っているノードからランダムに取得先を選択して取得する。この場合はノード内のデータキャッシュに保存して再利用する。

2.6 複製管理

データブロック及び構成データは複製を作成するが、複製とオリジナルには区別はなく、全て同じ扱いをする。複製は一定数(3～5 程度)を保つように定期的に調整する。ファイルの書き込みを行う場合にはファイルの複製全てを更新する。また、障害が発生した場合に備えて定期的に複製間で一貫性が保たれているかのチェックや、どの構成データからも参照されていない不要なデータが残っていないかのチェックも行う。

2.7 データの再配置

あるユーザが頻繁に利用する傾向にあるノードにそのユーザが利用するファイルの複製が配置されていると、ファイルアクセス時のオーバヘッドが少ない。VDFS ではそれぞれのファイルのアクセス頻度によって複製を利用頻度の高いノードに再配置することを行う。このために、各ノードは一定時間内のデータのアクセス回数を記録し、一定数を超えた場合には配置ノードを変更する。

3. 実装

VDFS は C++言語を用い、Linux 上で実装した。ファイルシステムとしてのインタフェースにはユーザ空間でのファイルシステム構築を容易にする FUSE([3])を用いた。

4. おわりに

現在、ノード管理、データアクセス、複製の管理まで実装を行った。基本的な動作は確認している。今後、データの再配置の部分の実装を行い、NFS[4]のような集中サーバ型の分散ファイルシステムや他のサーバレスの分散ファイルシステムとの性能比較を行っていきたい。

参考文献

[1]T. Anderson, M. Dahlin, J. Neefe, D. Patterson, D. Roselli and R. Wang; Serverless Network File Systems. 15th ACM Symposium on Operating Systems Principles(SOSP '95). pp.109-126, Dec. 1995.

[2]F. Dabek, M. Frans Kaashoek, D. Karger, R. Morris, and I. Stoica; Wide-area cooperative storage with CFS. 18th ACM Symposium on Operating Systems Principles(SOSP '01). pp.202-215, Oct. 2001.

[3]Filesystem in Userspace. http://fuse.sourceforge.net/

[4]R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyo; Design and implementation of the Sun Network Filesystem; USENIX Summer 1985. pp.119-130, June. 1985.