個人向け計算機環境における分散ストレージシステムの試作

宮澤 元
南山大学 数理情報学部

E-mail: miyazawa@it.nanzan-u.ac.jp

本稿では我々が現在実装中の個人向け計算機環境における分散ストレージシステムについて述べる。これは、ストレージ層とファイルシステム層の二層構造に基づく分散ファイルシステムのストレージ層にあたるものであり、ネットワークに接続された物理ディスク群を用いてファイルシステム層に対して仮想ディスクを提供する。プロトタイプシステムでは複数の物理ディスクにブロックをストライピングして格納する機能を実現した。

1 はじめに

パーソナルコンピュータの普及に伴い、一人の利用者が複数の計算機を使い分けて利用する機会が増えている。例えば、職場や自宅ではデスクトップPCを利用しつつ、出先や移動中にノートPCを利用するケースがこれに該当する。このように複数の計算機を使い分ける場合、利用者がどの計算機を使う場合でも作業を継続できるように、各計算機間で情報を共有する必要がある。このために従来からさまざまな方法が提案されている。例えばLocal Area Network (LAN) 上では、以前からSun NFS[8, 5] のような分散ファイルシステムを利用した情報共有が行われており、近年では、パーソナルコンピュータ向けに開発されたServer Message Block (SMB) といったファイル共有プロトコルも広く利用されている。

このようなファイル共有に基づく情報共有の方法は、各計算機が比較的広帯域なネットワークで相互に接続されていることを前提としているので、ノートPCのようにネットワークから切断されることがあったり、自宅と職場に置かれた計算機同士のように高速に通信できないような場合には効率的に動作させることが難しい。このような状況に対応するためにファイルキャッシュの技術を拡張したCoda[1] のようなシステムも存在するが、

- システムが扱う情報量が増大し、単一のファイルサーバで情報を集中管理するのが困難である
- ファイルサーバを複数用いる場合、管理・運用コストが増大する
- Codaのようにファイルキャッシュを拡張する手法では、ファイルサイズの大きなファイルを管理するコストが大きい[1]

などの理由で広く普及するには至っていない。

このような問題に対する解決策として、計算機間の情報共有システムを、共有仮想ディスク

[1] Codaの原型となったAndrew File System[6] のバージョン3.0以降では、初期のデザインと異なり、ファイルを64KB単位でキャッシュするような改良が加えられている。

を提供するストレージ層と、この上でさまざまなファイルサービスを実現するファイルシステム層とに分離して構成するアプローチ [3, 9, 7] が近年広く研究されている。これは、ネットワークによる分散性をストレージ層で吸収することにより、ファイルサーバによるファイルの集中管理コストと、キャッシュ管理コストを削減できるからである。

我々は、個人向けの計算機環境において、ネットワークを利用して柔軟な情報共有を行うための枠組みを実現することを目的として研究を行っており、ストレージ層とファイルシステム層の二層構造に基づく分散ファイルシステムを新たに設計し、実装中である [10]。本稿ではこのシステムのストレージ層となる分散ストレージシステムについて述べる。本システムは、ネットワークに接続された物理ディスク群から単一の仮想ディスクを構成し、これをファイルシステム層に提供するものである。分散性を吸収するとともに各ブロックについて複製を作ることによって冗長性をも確保することができる。

以下、2 節では本分散ストレージシステムの設計について述べる。システムの構成とともに、システムの動作についても説明する。プロトタイプの実装とこれを用いた簡単な実験について 3 節で述べる。4 節で関連研究を紹介し、5 節を本稿のまとめとする。

2　分散ストレージシステムの設計

この節では、本分散ストレージシステムの設計について、システム構成を述べる。分散性の吸収と冗長性の確保についてと、ブロックの読み出し・書き込み操作についても説明する。

2.1　システム構成

本分散ストレージシステムは、以下のように構成される (図 1)。

- ストレージインタフェース部
  ファイルシステム層からのブロック読み出し・ブロック書き込み要求を受け付ける。ファイルシステム層で管理される情報を受け付けるためのインタフェースもここに含まれるが、このインタフェースは現在設計中であり、詳細は決まっていない。

- ブロック通信部
  ストレージインタフェース部で受け付けた要求が他のホストが持つブロックに対するものだった場合、そのホストとの間でブロックの送受信を行う。

- 論理ブロック管理部
  論理ブロック番号と、そのブロックが存在するホストと物理ブロック番号の対応付けを管理する。各ブロックは複製されているのでこの対応は一般に一対多の対応となり、冗長性を確保できる。

- 物理ブロック管理部
  物理ディスクを管理し、ストレージインタフェース部やブロック通信部からの要求を受けてブロックを物理ディスクから読み書きする。

2.2　分散性の吸収

ストレージインタフェース部で受け取ったファイルシステム層からのアクセス要求中の論理ブロック番号は、論理ブロック管理部でホストと物理ブロックとの組に変換される。該当物理ブロックがローカルホストに存在している場合、そのブロックに関する要求が物理ブロック管理部に伝えられ、処理される。それ以外の場合、ブロック通信部が該当ブロックが存在しているホストと通信し、該当ブロックのアクセスを行う。

このとき、ブロック通信部が受信したブロックをローカルディスクにキャッシュすることもできる。この場合、ブロック通信部は物理ブロッ

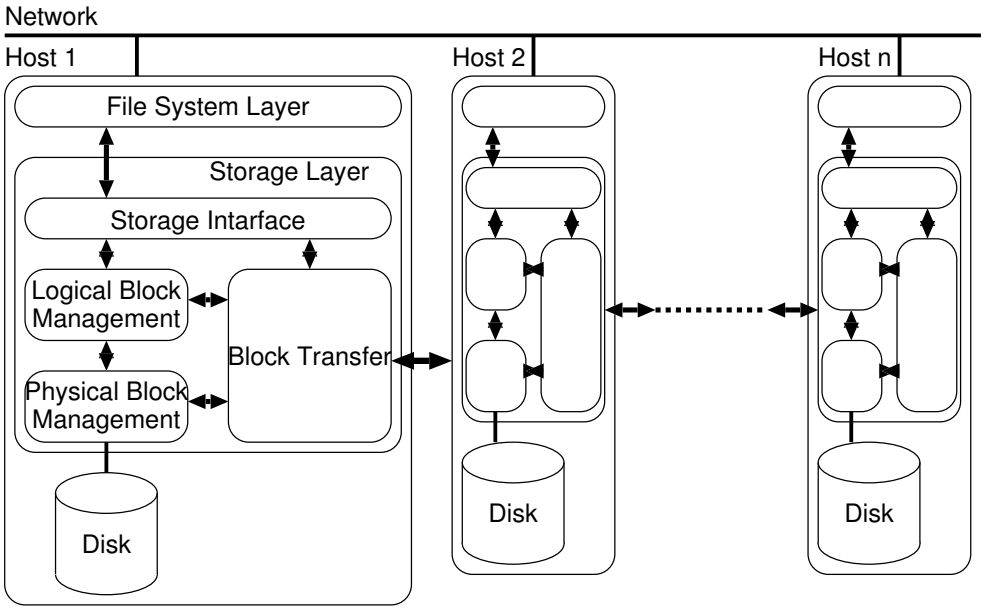

図 1: 分散ストレージシステムの構成

ク管理部に受信ブロックのキャッシュを依頼する。このキャッシュは、性能向上のために一時的に保持される場合と、次節で述べる冗長性の確保のための複製ブロックとして用いられ、長期的に保持される場合がある。

2.3 冗長性の確保

ある論理ブロックに対して複数の物理ブロックを対応させることにより、冗長性を確保する。通常、これらの物理ブロックの内容は全く同じであり、それぞれを異なるホストに配置することによって、あるホストが障害などでダウンした場合でもサービスの継続が可能となる。

性能向上の観点や、利用しているホストがネットワークから切断されるときのことなどを考えると、複製ブロックの一つは利用しているホスト上に保持されることが望ましい。それ以外の複製ブロックも、そのブロックを利用する可能性の高いホストに配置するべきである。このような複製ブロック配置を決定するにあたっては、ファイルシステムの利用状況やファイルの所有者状況など、ファイルシステム層から受け取る情報を利用する。

2.4 ブロックの読み書き

ある論理ブロック読み出し要求に対して、複数の物理ブロックが対応する場合、利用しているホストからネットワーク的に最も近いホストに対して読み出し要求を出す。ホスト障害などでアクセスできない場合、他の複製ブロックを順次要求していく。また、モバイルホストなどがネットワークから切断されている場合、他のホストと通信することができず、必要なブロックを読み出せないことがありうる。このような場合には、Coda[1] の hoarding と同様の技術を用いて、必要なブロックをあらかじめモバイルホストのディスクに保持しておく必要がある。

論理ブロックを書き込む場合、このブロック

に対応する複製ブロックを全て更新する必要がある。しかし、複製ブロックを保持しているホストが障害などでダウンしている場合、その複製ブロックを更新することができない。障害でダウンしている場合、障害を復旧して再起動する際にディスクをチェックして、複製ブロックを最新のものに更新することができる。モバイルホストなどがネットワークから切断されている場合にも、ホスト障害と同様、他のホストと通信できなくなるのでモバイルホストの持つ複製ブロックを更新できない。この場合にも、障害から復旧したときと同様、ネットワークに再接続した時に複製ブロックを更新する必要がある他、モバイルホスト自身が更新したブロックを他のホストに複製する必要がある。このとき、あるブロックが 2 台以上のホストで更新され、更新が衝突することがありうる。このような衝突をある程度自動的に解決する方法も提案されている [2] が、何らかの形でユーザの判断をあおぐような機構も用意すべきであろう。

3 実装および実験

現在、2 節の設計に基づき、Intel CPU ベースの PC を用いて Linux 上に本分散ストレージシステムの実装を行っている。現在までに、機能的な制限があるものの、ユーザレベルで動作するプロトタイプシステムが完成している。

3.1 プロトタイプシステムの実装

プロトタイプシステムでは、ネットワーク上での動作確認を優先し、2 節の設計を簡略化した形で、ユーザレベルサーバとして実現している。

- ストレージインタフェース部
  ストレージインタフェース部は、本来ブロック型デバイスドライバとして実現すべきであるが、プロトタイプシステムではブロック通信部と統合している。クライアントプロセスは、TCP/IP を用いたローカルなプロセス間通信を行ってストレージインタフェース部にアクセスする。また、ファイルシステム層からのヒント情報を受け取るインタフェースは実装されていない。

- ブロック通信部
  通常のネットワークサーバと同様に他の計算機と TCP/IP で通信することによりブロックの送受信を行う。現在の実装では、1 論理ブロック毎に物理ブロックとのマッピングを行っており、ブロック通信も 1 ブロック単位に行われる。

- 論理ブロック管理部および物理ブロック管理部
  プロトタイプシステムでは、論理ブロック管理部と物理ブロック管理部は、論理ブロックと物理ブロックのマッピングを行うモジュールとしてひとまとまりに実装されている。物理ブロックはブロックが存在する計算機とその計算機上での物理ブロック番号の組として表現される。現在のところ、論理ブロックと物理ブロックのマッピングは、システムを構成する計算機群でストライピングを行うように固定されており、ブロックに冗長性を持たせるなど、マッピングを動的に変更することは出来ない。

3.2 プロトタイプシステムを用いた実験

プロトタイプシステムを用いて、1000Base-T ネットワークで接続された PC(CPU: Pentium4 2.4GHz, Memory: 1GB)6 台に Linux(カーネルバージョン 2.4.20) をインストールした環境で簡単な性能測定実験を行った。この実験では 1 ブロックを 512 バイトとして 1000 ブロックのアクセスを 100 回繰り返すのに要する時間を計測した。プロトタイプシステムは、ひとつのクライ

アントプロセスから出されたアクセス要求を 6 台の PC にストライピングして処理する。比較対象として TCP アクセスとデバイスアクセスの二種類を用意した。前者は、本システムからストライピング機能を外したもので、ひとつのクライアントプロセスが指定した 1 台のホスト上のサーバプロセスに対してアクセスを行う。後者は Linux システムコールを用いてディスクデバイスをオープンし、読み書きを行うものである。実験結果を表 1 に示す。この表では、TCP アクセスの結果はクライアントプロセスとサーバプロセスが同一ホストに存在する場合 (ローカル) と異なるホストに存在する場合 (リモート) に分けて表示されている。

表 1: 1000 ブロック ×100 回のアクセスに要する時間 (秒)

	読み出し	書き込み
本システム	112.44	93.98
TCP アクセス		
ローカル	4.06	0.85
リモート	4.97	1.19
デバイスアクセス	1.10	0.09

ホスト間のブロック転送を 1 ブロックごとに行っている影響で、本システムの性能は TCP アクセスに比べて非常に悪いことが分かる。また、全般に読み出しに比べて書き込みの方が所要時間が短いが、これは書き込みキャッシュが影響したものだと考えられる。

4 関連研究

複数の計算機に接続された物理ディスク群を用いて仮想ディスクを構成するようなシステムは、これまで数多く提案されている。

Petal[3] と Frangipani[9] はファイルシステム層とストレージ層の二層構造をとる分散ファイルシステムである。前者が分散ストレージシステムであり、後者が前者の上で動作するファイルシステムにあたる。分散性の吸収やブロックの複製による冗長性の確保は Petal で行われているが、ファイルシステムである Frangipani の管理情報を利用するようなことは行われていない。

PersonalRAID[7] は、互いにネットワークで接続されていない計算機同士の間で共有仮想ディスクを構成する分散ストレージシステムである。ブロック操作のログを記録したリムーバブルディスクを持ち運ぶことにより、

ファイル共有を透明に行うことができる。しかし、共有仮想ディスクを実現するためにネットワーク接続を利用することはない。

Boxwood[4] は、ファイルシステム層により高機能なインタフェースを提供することを目的とした分散ストレージシステムである。ディスクブロックを操作する低レベルなインタフェースではなく、B 木のような抽象度の高い操作を提供することにより、ファイルシステム層によって直接ブロックを管理することによって生じる複雑さを軽減することができる。Boxwood のアプローチは、ストレージ層がディスクブロック操作以外のインタフェースを備える点で我々の研究と共通しているが、ストレージ層の動作に対してファイルシステム層がヒントを与えるようなことは考慮されていない。

5 おわりに

我々はストレージ層とファイルシステム層の二層構造に基づく分散ファイルシステムの設計・実装を進めており、本稿ではこのシステムのストレージ層にあたる分散ストレージシステムについて述べた。プロトタイプシステムでは複数の物理ディスクにブロックをストライピングして格納することができる。しかし、プロトタイプシステムでは性能的にも不十分である。今後

はファイルシステム層と合わせて設計・実装をより洗練させる。

謝辞

この研究は、2004 年度 南山大学パッヘ研究奨励金 (Pache Research Subsidy)I-A-1 の助成を受けています。

参考文献

[1] James J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. In *Proceedings of 13th ACM Symposium on Operating Systems Principles*, pages 213–225, October 1991.

[2] Puneet Kumar and M. Satyanarayanan. Flexible and Safe Resolution of File Conflicts. In *Proceedings of the Usenix Winter 1995 Technical Conference on Unix and Advanced Computing Systems*, January 1995. available via ftp (CMU-CS-94-214).

[3] Edward K. Lee and Chandramohan A. Thekkath. Petal: Distributed Virtual Disks. In *Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS VII)*, pages 84–92, October 1996.

[4] John MacCormick, Nick Murphy, Marc Najork, Chandramohan A. Thekkath, and Lidong Zhou. Boxwood: Abstractions as the foundation for storage infrastructure. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI'04)*, pages 105–120, December 2004.

[5] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. Design and Implementation of the Sun Network Filesystem. In *Proceeding of the Summer 1985 USENIX Conference*, pages 119–130, Portland OR (USA), June 1985. USENIX.

[6] M. Satyanarayanan, John H. Howard, David A. Nichols, Robert N. Sidebotham, Alfred Z. Spector, and Michael J. West. The ITC Distributed File System: Principles and Design. In *Proceedings of the 10th ACM Symposium on Operating Systems Principles*, pages 35–50, December 1985.

[7] Sumeet Sobti, Nitin Garg, Chi Zhang, and Xiang Yu. PersonalRAID: Mobile Storage for Distributed and Disconnected Computers. In *Proceedings of the FAST 2002 Conference on File and Storage Technologies*, January 2002.

[8] Sun Microsystems, Inc. NFS: Network File System Protocol Specification. RFC 1094, Network Information Center, SRI International, March 1989.

[9] Chandoramohan A. Thekkath, Timothy Mann, and Edward K. Lee. Frangipani: A Scalable Distributed File System. In *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles*, pages 224–237. ACM, October 1997.

[10] 宮澤 元. ファイルシステム情報を利用する分散ストレージシステム. **アカデミア (南山大学紀要) 数理情報篇**, 第 3 巻:47–52, 3 月 2003 年.