データベースシステム 128－27
（２００２．７．１８）

自律ディスクを用いたWebサーバにおける負荷偏りの影響

花井 知広 * 横田 治夫 †,*

* 東京工業大学 大学院 情報理工学研究科 計算工学専攻

† 東京工業大学 学術国際情報センター

〒 152-8552 東京都目黒区大岡山 2-12-1

hanai@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp

概 要

我々は、高いスループットとアベイラビリティを持つ Web サーバの管理コストを低減させるためのアプローチとして、自律ディスクを用いたクラスタ化 Web サーバを提案してきた。本稿では提案構成の負荷に偏りがある場合の性能を測定し、NFS および Apache の Proxy Throughput 機能を用いてクラスタ化 Web サーバを構成した場合との性能比較を行い、偏りのある場合の提案手法の有効性を示す。さらに HTTP インターフェースを持つ自律ディスクの構成方法として、Apache が直接オブジェクトにアクセスするような方式を提案し、同様の比較評価を行いその有効性を示す。

キーワード： 自律ディスク, Web サーバ, クラスタ, 負荷分散, 偏り制御

Influence of load skew on a Web Server constructed from Autonomous Disks

Tomohiro HANAI* Haruo YOKOTA†,*

* Department of Computer Science Graduate School of Information Science and Engineering

Tokyo Institute of Technology

† Global Scientific Information & Computing Center Tokyo Institute of Technology

2-12-1 Oh-Okayama, Meguro-ku Tokyo, 152-8552 JAPAN

hanai@de.cs.titech.ac.jp, yokota@cs.titech.ac.jp

Abstract

We have proposed a Web server cluster constructed from Autonomous Disks to realize a high throughput and high availability Web server with less administration cost. In this paper, we analyze the performance with load skew, and compare performance with other methods such that constructed from NFS or Apache Proxy Throughput, to indicate effectiveness of the proposal method. Furthermore, as a composition method of an Autonomous Disks with a HTTP interface, we propose a method that the Apache directly access to objects. We also analyze its performance to indicate effectiveness of the approach.

KEYWORD: Autonomous Disks, Web Server, Cluster, Load Balancing, Skew Handling

1 はじめに

今日、Web の急速な普及によって、Web サーバにはますます高いスループットとアベイラビリティが要求されるようになってきている。同時に、コンテンツ容量の増加に伴いストレージ管理にも工夫が必要となっている。これらの要求を満たすためには、クラスタ化 Web サーバを構成することが一般的であるが、システムの規模が大きくなるにつれて管理コストも増大することが問題になっている。

我々は、ディスク装置内の制御用プロセッサとキャッシュ用のメモリを利用し、装置内で高度な機能 (負荷分散、故障対策、障害回復など) を実現し管理コストを低減するためのアプローチとして、自律ディスクを提案してきた [1, 2, 3]。この自律ディスクは、ネットワークに複数接続されクラスタを構成し、ホストから透過的に負荷分散・故障対策・障害回復を行うものである。そのために、分散ディレクトリ、ECA ルール、ディスク間通信機能、トランザクション機能を備えている。そしてその有効性を示すために Java で模擬自律ディスクの試作を行ってきた。

これまでの模擬自律ディスクの外部インターフェースは、Java のオブジェクトシリアライズ機構をベースにしたものであった。しかし、より汎用性を持たせるために、広く用いられている既存のプロトコルの利用を考える必要があると考え、その 1 つのアプローチとして HTTP を用いた外部アクセスを検討してきた [4]。様々な構成方法が考えられるが、まず簡単にできる方法として Web サーバとして Apache の利用を考えた。つまり、模擬自律ディスクを Apache から利用する方法である。これについて基本性能の評価を行ったが、負荷が偏った場合などのもとでの評価は不十分であった。

そこで本稿ではまず、Apache から模擬自律ディスクを利用する方式で、負荷が偏った場合の性能評価を行う。また、自律ディスクは負荷分散、故障対策、障害回復といった機能を実現するためのものであるため、Apache に分散ディレクトリ、ECA ルール、ディスク間通信、トランザクション機能を組み込むことが可能であれば、それを HTTP インターフェースを持つ自律ディスクとみなすこともできる。その一歩として、ストリーム自体を Apache から直接アクセスするように変更した構成を提案し、その基本性能を評価する。

2 自律ディスク

はじめに自律ディスクの特徴を説明する。自律ディスクはネットワーク環境でクラスタを構成することを前提としている。ホストはデータにアクセスするためにクラスタ内の任意のディスクに要求を発する。クラスタ内のディスクはディスク間の局所的な通信を行うことで、協力してホストからの要求に対処する。このような前提のもとで、自律ディスクは以下のような性質を持つ [1]。

- データ分散
- ホストからの均質的なアクセス
- 同時実行制御
- 偏り制御
- 耐故障性
- 異種性

これらの性質の主な長所は、ホストからの透過性とシステムのスケーラビリティである。データ分散の仕方、偏り制御の方法、耐故障性、異種性等に関してホストは関与しないため、分散ディスク制御に対するホストのオーバーヘッドをかなり減少させることができる。

3 Apacheから模擬自律ディスクを利用する場合

2 章で述べた自律ディスクを用いてクラスタ化 Web サーバを構成することで、以下の利点を得られると考えられる。

- 自律ディスクがオブジェクトを各ノードに均等に分散して配置するので、ノード間の負荷の偏りを減少させることができる。また、特定のオブジェクトにアクセスが集中した場合にも動的

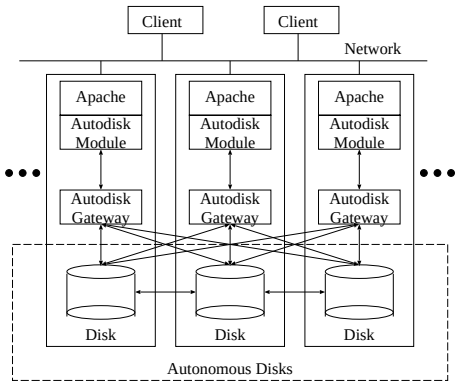

図 1: 自律ディスクを用いた Web サーバ

な偏り除去が行われ、負荷が分散される。これにより、管理者はオブジェクトの配置を考える必要がなくなり、アクセス偏りの時間的な変化に対して迅速に対応できる。

- ディスクが故障した場合にも、自律ディスクが持つバックアップ機構を利用してサービスを継続することができる。また、ダーティリードを許可する場合はバックアップディスクからも読み出すことができ、さらに負荷を分散させることができる。

この Web サーバを構成する上では、以下のような条件を仮定している。

- CGI などを用いて動的にオブジェクトを生成することは考えず、静的オブジェクトのみを扱う。
- クラスタ内の各ノードが HTTP リクエストを受け、コンテンツをサービスする。
- HTTP リクエストはクラスタ内の各ノードに均等に与えられるものとする。これはラウンドロビン DNS 等の手法を用いることで実現できる。

3.1 システム構成

この自律ディスクを用いた Web サーバは以下の要素から構成されている。その構成を図 1 に示す。

Apache Web Sever HTTP のハンドリングを行う [5]。この Apache には自律ディスクゲートウェイと通信する機能がモジュールとして組み込まれている。このモジュールは Apache 自律ディスクモジュールと呼ばれる。設定ファイルで指定したディレクトリ以下へのアクセスを自律ディスクへのオブジェクト要求とみなし、処理を行う。

自律ディスクゲートウェイ Apache 自律ディスクモジュールが自律ディスクにアクセスするためのインターフェースである。Apache 自律ディスクモジュールからリクエストされたオブジェクトを自律ディスクにリクエストし、そのオブジェクトを得て Apache 自律ディスクモジュールに返す。自律ディスクへのリクエスト先はルールによって指定することができる。

自律ディスク オブジェクトを格納し、そのディレクトリ情報を持つ。現在、Java で模擬実装が行われている [2]。

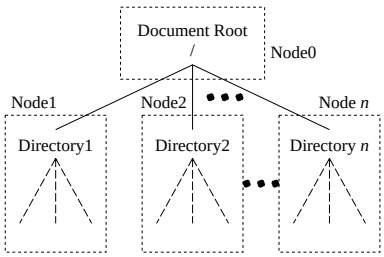

図 2: 比較対象システムでのオブジェクトの分散配置

3.2 比較対象システム

性能を測定する上で、比較対象としてオブジェクトをクラスタ内に分散配置する以下ような 2 つのシステムを考える。これらのシステムでは図 2 のようにディレクトリごとにオブジェクトを各ノードに分散させてデータを配置する。

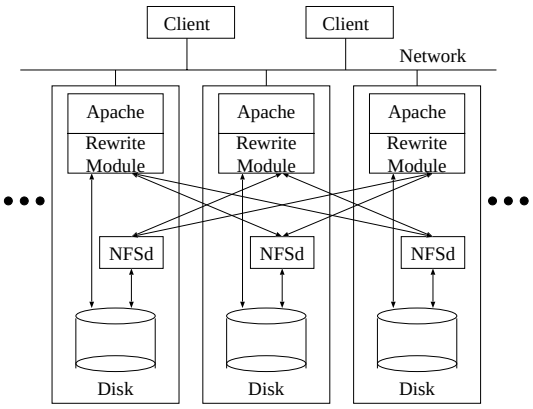

図 3: NFS を用いた Web サーバ

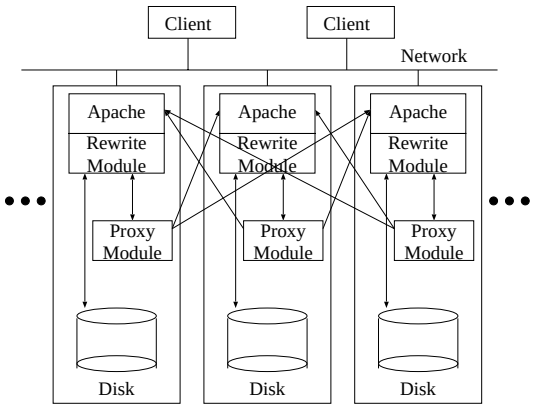

図 4: rewrite モジュールの Proxy Throughput 機能を用いた Web サーバ

3.2.1 NFS を用いたオブジェクトの分散配置

このシステムではオブジェクトの読み出しに NFS プロトコルを用いる。その構成を図 3 に示す。この構成では Web サーバに対する HTTP リクエストは以下の流れで処理される。rewrite モジュールとはルールを用いて URL を内部的に書き換える Apache のモジュールである。

1. Apache は HTTP リクエストを受けると、URL からクラスタ内のオブジェクトを要求しているかどうかを判定する。要求しているならば rewrite モジュールに URL を渡す。

2. rewrite モジュールはマッピングが書かれた設定ファイルに従って URL に含まれるディレクトリ名からそのオブジェクトを持つノードを求める。

3. オブジェクトが自ノードにある場合は Apache がそのオブジェクトをクライアントに送信する。他のノードにある場合はオブジェクトを NFS プロトコルを用いて読み出し、クライアントに送信する。

3.2.2 Apache の rewrite モジュールの Proxy Throughput 機能を用いたオブジェクトの分散配置

この手法はオブジェクトの読み出しに Apache の rewrite モジュールと proxy モジュールを用いるものである。簡単のために以下この手法を APT と呼ぶ。その構成を図 4 に示す。proxy モジュールはプロキシー機能を実現する Apache のモジュールである。この構成では Web サーバに対する HTTP リクエストは以下の流れで処理される。

1. Apache は HTTP リクエストを受けると、URL からクラスタ内のオブジェクトを要求しているかどうかを判定する。要求しているならば rewrite モジュールに URL を渡す。

2. rewrite モジュールはマッピングが書かれた設定ファイルに従ってディレクトリ名からそのオブジェクトを持つノードを求める。

3. オブジェクトが自ノードにある場合は Apache がそのオブジェクトをクライアントに送信する。他のノードにある場合はそのノード名と URL を proxy モジュールに渡す。

4. proxy モジュールは渡された情報を元にオブジェクトを持つノードに HTTP リクエストを行い、オブジェクトを得てクライアントに送信する。

3.3 測定条件

自律ディスクを用いたWebサーバと、2つの比較対象システムに対して性能を測定し、比較を行う。測定に使用した環境は表1の通りである。クライアントもクラスタの各ノードと同じ性能を持っており、クラスタ–クライアント間もギガビットイーサネットを用いて接続されている。[4]で性能測定を行った際に、クライアント側がボトルネックになっていたので、今回はクライアントを2台とした。

測定方法は以下の通りである。

- クラスタ内の各ノードに対してHTTPリクエストを発行し、クラスタ全体に対するスループットを測定する。リクエストは一定の同時接続数で60秒間行う。
- 一定サイズのオブジェクトをクラスタ全体で1024個格納する。
- どのノードにリクエストが行われるかはランダムである。
- どのオブジェクトをリクエストするかもランダムである。
- これらの条件で測定するためにベンチマークソフトウェアとしてhttp_load[6]を複数台のクライアントから同時にベンチマークを行えるように変更して使用した。
- 自律ディスクを用いた手法の場合、自律ディスクゲートウェイにおいてリクエスト先ディスクを選択するルールは、全ノードからランダムに選ぶ方法とする。

表1: 実験環境

ノード数	6台
CPU	Intel Pentium III 933 MHz
メモリ	PC133 SDRAM 256MB
HDD	Seagate Barracuda IV 20.4GB 7200rpm
OS	Linux 2.2.17
ネットワーク	1000BASE-SX
Java環境	IBM JDK 1.3.0
Apache	Version 1.3.22
クライアント台数	2台

これらの条件で以下の測定を行った。

1. オブジェクトサイズの変化に対するスループットの変化
2. ノード間のオブジェクト格納数の偏りに対するスループットの変化

2については本来はアクセス負荷の偏りに対してのスループットの変化を測定すべきである。しかし現在の自律ディスクの負荷均衡化はディスク使用量を基に行うよう実装されており[7]、その測定が困難であるためにこちらを測定した。アクセス頻度の偏りへの対策も検討中であり、詳しくは[8]を参照されたい。

オブジェクト格納数の偏りの生成には以下の式

$$a_i = \frac{1}{i^\theta}$$

で与えられるzipf分布をもとに、ノード i に格納されているオブジェクト数 $d(i,\theta)$ を

$$d(i,\theta) = n \times \frac{a_i}{\sum a_i}$$

とした。ここで θ は偏りの度合いを表すパラメータである。$\theta = 0$ の時は偏りはなく、$\theta = 0$ が大きくなるほど偏りも大きくなる。また n はクラスタ全体でのオブジェクト数であり、ここでは $n = 1024$ である。

また、自律ディスクを用いたWebサーバではノード間のオブジェクト格納数の偏りは自律的に除去されるので、オブジェクト格納数に偏りがある場合の測定は行わない。

3.4 測定結果

まず、2つの比較対象システムについての測定結果を3Dグラフにしたものを図5(NFSの場合)と図6(APTの場合)に示す。このグラフより、偏りがない ($\theta = 0.0$) 場合を抜き出し、同条件で自律ディス

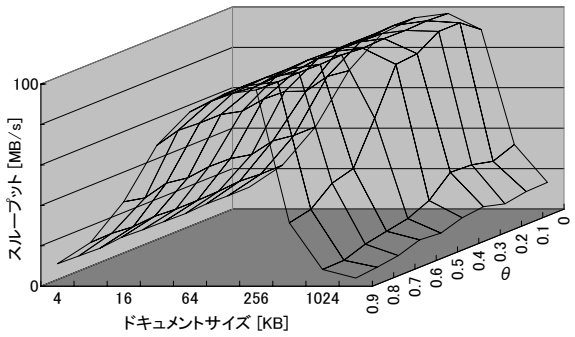

図 5: NFS

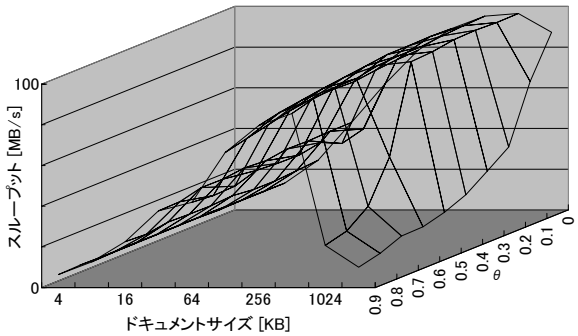

図 6: APT

クを用いた Web サーバに対して測定したものとを比較したものが図 7 である。[4] での測定結果よりも NFS や APT のスループットがかなりよくなっているが、これは [4] の測定においてはクライアントがボトルネックになってしまっていたためである。

この図からわかる通り、初めはオブジェクトサイズが大きくなるしたがって各手法ともスループットが増加している。これは、オブジェクトサイズが大きくなるにしたがって Web サーバに対する接続の確立や HTTP のハンドリングにかかる時間がオブジェクトの転送時間比べて相対的に短くなるからであると考えられる。また、各手法ともオブジェクトサイズがある値を超えた後はスループットが低下している。これは、オブジェクトサイズが大きくなるにつれて各ノードでのオペレーティングシステムのディスクキャッシュのヒット率が低下するからであると考えられる。NFS が APT よりスループットが早く低下する理由として、NFS プロトコルでは他の

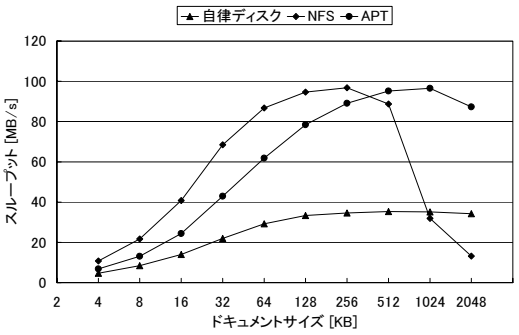

図 7: オブジェクトサイズとスループット (偏りなし)

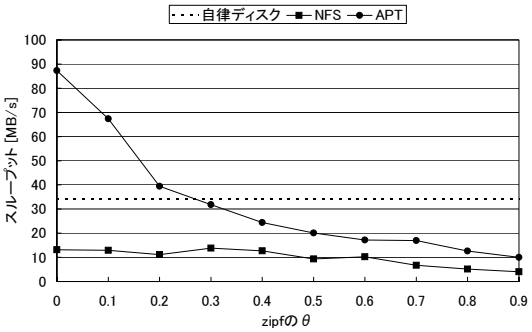

図 8: 偏りとスループット (オブジェクトサイズ =2048KB)

ノード内のオブジェクトもメモリにキャッシュするのに対して、APT では他のノードのオブジェクトはキャッシュされない。そのため、APT の方が自ノードのディスクに対するキャッシュメモリ容量を大きく取ることができる。よって、ネットワークが高速ならば APT の方が NFS よりもオブジェクトサイズの増加に対してスループットが低下しにくくなる。

次に、オブジェクトサイズが 2048KB の時の偏りの大きさとスループットの関係を図 5,6 から抽出したものを図 8 に示す。自律ディスクを用いた Web サーバでは偏りが発生した場合には自律的に偏りが除去されるので、長時間にわたって偏りが発生することはない。そこで自律ディスクを用いた Web サーバでは偏りがないときのスループットを点線で示してある。NFS と APT は偏りが大きくなるにつれてスループットが低下している。このグラフより、偏

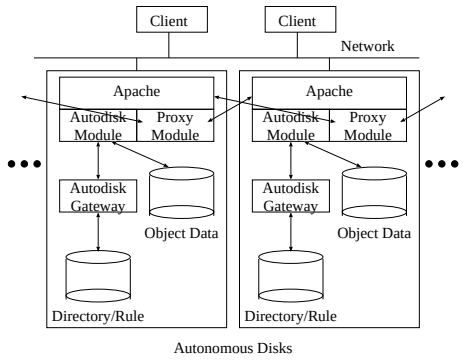

図 9: Apache が直接オブジェクトにアクセスする Web サーバ

りがないときは自律ディスクを用いた Web サーバを大きく上回っている APT も、偏りが 0.25 程度以上になるとスループットが自律ディスクを用いた Web サーバを下回ってしまうことがわかる。

3.5 Apache から直接オブジェクトにアクセスする場合

偏りがない場合に模擬自律ディスクを用いた Web サーバで性能が伸びないのは、大きなオブジェクトの読み出し時の自律ディスクや自律ディスクゲートウェイでのオーバーヘッドが大きくなっているからだと考えられる。そこで性能を改善するために、オブジェクト自身は自律ディスクの中に格納せず、通常のファイルシステム上にファイルとして格納する。そして自律ディスクの分散ディレクトリをルールでその格納位置を管理する。これにより、自律ディスクの持つべき機能を満たしながら、データ転送の効率を上げることができる。

3.6 システム構成

このシステムの構成図を図 9 に示す。Web サーバに対する HTTP リクエストは以下の手順で処理される。

1. Apache は HTTP リクエストを受けると、URL から自律ディスク内のオブジェクトを要求しているかどうかを判定する。要求しているならば Apache 自律ディスクモジュールにリクエストの処理を任せる。
2. Apache 自律ディスクモジュールは URL から自律ディスク内でのストリーム ID を取り出し、自律ディスクゲートウェイにリクエストを発行する。
3. 自律ディスクゲートウェイは Apache 自律ディスクモジュールから自律ディスク内のオブジェクトのストリーム ID を受け取り、自律ディスクに Retrieve コマンドを送信する。
4. 自律ディスクは Retrieve コマンドを受けて分散ディレクトリをトラバースし、そのオブジェクトをどのノードが持っているかを自律ディスクゲートウェイに送る。
5. 自律ディスクゲートウェイは自律ディスクからリクエストの結果を受け取り、Apache 自律ディスクモジュールに送信する。
6. Apache 自律ディスクモジュールは自律ディスクゲートウェイからオブジェクトをどのディスクが持っているかを受信する。もし自ノードにあるならばそれを読み出して HTTP リクエスト元に送信する。
7. 自ノードになかった場合はオブジェクトを持っているディスクから Proxy Throughput 機能を利用してオブジェクトを得て、それを HTTP リクエスト元に送信する。

3.7 測定結果

以上のような改良を行って前章と同様に測定を行った結果が図 10 である。自律ディスクと APT を組合せた Web サーバは自律ディスクのみを用いた Web サーバに比べ、大きくスループットが改善されており、APT の場合に似た軌跡を描いている。APT と自律ディスクと APT を組合せた Web サーバの差が、自律ディスクゲートウェイにおけるオーバーヘッドと模擬自律ディスクにおけるルール処理や分散ディ

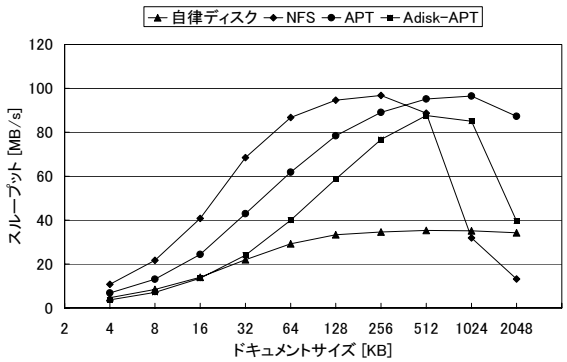

図 10: Apache が直接オブジェクトにアクセスする Web サーバのスループット

レクトリ操作のオーバーヘッドであると考えられる。

4 まとめと今後の課題

本稿では、クラスタ化 Web サーバにおいてノード間の負荷の偏りが性能に与える影響について測定し、一定量以上負荷の偏りが存在する場合には自律ディスクを用いた Web サーバが有効であることを示した。

また、自律ディスクを用いて構成された Web サーバを HTTP インターフェースを持つ自律ディスクとみなし、オブジェクト自身は OS のファイルシステム内に直接格納し、そのディレクトリ情報を模擬自律ディスクで管理する新たなアーキテクチャを提案した。これによりオブジェクト自身を読み出す際のオーバーヘッドを、オブジェクトを模擬自律ディスク内に配置した場合に比べて大きく減らすことができる。

今後の課題として、本稿ではノード間の負荷の偏りとして各ノードのディスク使用量の偏りを仮定したが、本来は各オブジェクトに対するアクセス頻度の偏りなども考慮してノード間の負荷の偏りを扱う必要がある。現在の模擬自律ディスクは各オブジェクトに対するアクセス頻度の偏りの均衡化を行えないため、これを実装する必要がある。

また、オブジェクト自身は OS のファイルシステム内に格納し、そのディレクトリ情報は模擬自律ディスクで管理する場合、実際にどのようにしてマイグレーションを行うかはまだ規定されていない。模擬自律ディスクに大きなオブジェクトを格納した場合は、それが分割されて格納されるが、OS のファイルシステム上に配置した場合は、マイグレーションの粒度が荒くなることが考えられる。その際に精度良く負荷均衡化を行えるのかを確認する必要がある。

謝辞

本研究の一部は，文部科学省科学研究費補助金基盤研究 (12680333, 13224036, 14019035) および情報ストレージ研究推進機構 (SRC) の助成により行なわれた．

参考文献

[1] Haruo Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pages 441–448, Nov. 1999.

[2] 安部洋平 , 横田治夫. Java による耐故障ネットワークディスクのルール処理の実装. In **第 *11* 回データ工学ワークショップ論文集**, *DEWS2000 3B–2*. 電子情報通信学会データ工学研究専門委員会, 2000.

[3] 阿部 亮太 , 横田 治夫. 自律ディスクにおける故障時の動作とそのルール処理の実装. In **第 *12* 回データ工学ワークショップ論文集**, *DEWS2001 2B–3*. 電子情報通信学会データ工学研究専門委員会, 2001.

[4] 花井知広 , 横田治夫. 自律ディスクを用いた web サーバーの構成. In **第 *13* 回データ工学ワークショップ論文集**, *DEWS2002 C3–4*. 電子情報通信学会データ工学研究専門委員会, 2002.

[5] The Apache Software Foundation. Apache http server. http://httpd.apache.org/.

[6] ACME Laboratories. http_load - multiprocessing http test client. http://www.acme.com/software/http_load/.

[7] 伊藤大輔 , 横田治夫. 自律ディスク上の分散ディレクトリの負荷均衡機構を用いたクラスタ再構成. In **第 *13* 回データ工学ワークショップ論文集**, *DEWS2002 C3–3*. 電子情報通信学会データ工学研究専門委員会, 2002.

[8] 渡邊 明嗣 , 横田 治夫. 分散ディレクトリ偏り制御とシステム再構成を統合する再配置制御. In **情処学会研究会報告, データベースシステム** *DBS-128-1*, 2002. 投稿中.