社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

[Full Paper]

ルールを含むメタデータによる柔軟なコンテンツ管理

山口　宗慶† 　渡邊　明嗣† 　小林　大† 　田口　亮†† 　上原　年博††
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; 152-8552 東京都目黒区大岡山 2-12-1
†† NHK 放送技術研究所; 157-8510 東京都世田谷区砧 1-10-11
††† 東京工業大学 学術国際情報センター; 152-8550 東京都目黒区大岡山 2-12-1
E-mail: †muu@de.cs.titech.ac.jp, ††{aki,daik}@de.cs.titech.ac.jp, †††{taguchi.r-cs,hayashi.n-gm}@nhk.or.jp,
††††yokota@cs.titech.ac.jp

あらまし　近年，扱われるデータ量が増大し，それに伴いデータ管理コストの増加が問題になってきている．特に，扱いの異なる様々なコンテンツが存在するため，個々のコンテンツの特徴を考慮した管理が容易でないという問題がある．本研究では，各コンテンツに対応したメタデータとして ECA ルールで処理を記述することによって，個々のコンテンツに合った管理を行う手法を提案する．ECA ルールを用いることでストレージの状態を考慮した処理記述ができる．また，ECA ルール発火制御のために，ディレクトリ構造に工夫を加える．高機能ストレージとして提案されている自律ディスクへ提案手法を実装し，そのフィージビリティを示す.
キーワード　コンテンツマネジメント, メタデータ管理, 並列・分散 DB

Flexible Contents Management by Metadata Containing Rule

Munenori YMAGUCHI†, Akitsugu WATANABE†, Dai KOBAYASHI†, Ryo TAGUCHI††, Toshihiro UEHARA††, and Haruo YOKOTA†††

† Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†† NHK Science & Technical Research Laboratories
1-10-11 Kinuta Setagaya Tokyo, 157-8510 Japan
††† Global Scientific Information and Computing Center, Tokyo Institute of Technology
E-mail: †muu@de.cs.titech.ac.jp, ††{aki,daik}@de.cs.titech.ac.jp, †††{taguchi.r-cs,hayashi.n-gm}@nhk.or.jp,
††††yokota@cs.titech.ac.jp

Abstract　The treated volume of data increases, and an increase in the data management cost becomes a problem along with it in recent years. Especially, there is a problem that management that considers the feature of individual contents is not easy because contents treat different and a variety of exist. In this research, it proposes the technique for doing the management suitable for individual contents by describing processing by the ECA rule as meta data corresponding to each contents. The processing description to consider the state of storage by using the ECA rule can be done. Moreover, the device is added to the directory structure for the ECA rule ignition control. The proposal technique is mounted on the autonomous disk proposed as high performance storage, and the feasibility is shown.
Key words　Contents management, metadata management, parallel and distributed DB

1. 序　　論

ストレージシステムで扱われるデータ量と，データの種類や利用方法は日々増加しており，データの管理するコストが高くなってきている．特に重要なデータはバックアップとして何世代も保存され，雪だるま式に膨らむのみで，決して減ることは

ない．また，データの種類や利用法も増加してきており，同じ種類のデータでもそれぞれのデータの使われ方や，その価値によって扱いが異なることが多々ある．

そんな中，様々なデータの管理を自動化していきたいといった要求がある．特に学校や企業における大規模なストレージシステムにおいてはデータの増加による管理コストは問題であり，様々なデータ管理の自動化の要求も大きくなっている．

それらの要求を満たすための効率よいデータ管理方法として，最近は情報ライフサイクルマネジメント（Information Lifecycle Management:ILM）[1] という考えが注目されている．

ILM はその名の通り，データが一連のライフサイクルを行うことに注目し，その時間による価値の変化に対して最適な管理を行おうというものである．一般的なデータは生成・活用・保管・破棄という一連のライフサイクルを辿ると考えられている．データのアクセス頻度は，一般的には生成・活用の段階で最も高く，保存の段階で最も低くなると言われている．

アクセス頻度によって行うような ILM の一部として，Hierarchical Storage Management（HSM）[2], [3] がある．HSM は特に活用から保存の段階にかけての自動化を達成している．これは例えば高いスループットの高速ストレージと安定している低速ストレージを用意し，データのアクセス頻度から高速ストレージから低速ストレージへ移行するといったものである．これによって，高速のストレージにはアクセス頻度の高いデータのみが存在することとなり，高速ストレージの容量の節約になると共に高速に必要なデータを探し出すことができるようになる．最近の HSM の一例としては enigma [4] などがあり，製品として世の中に出ている．

しかし，HSM はアクセス頻度や過去のアクセス履歴の情報を用いているのみである．データにはアクセス情報だけではなく，データの重要度を始めとする個々のデータに対する情報が存在するが，HSM ではそれを活かしきってはいない．また，データをどこに配置するかというところだけに主眼が置かれているので，行うことが可能な管理動作もストレージ間の移動のみである．先述の通り，管理動作の自動化を行いたいことは様々であり，現在の HSM で満たせていることは一部のみである。

そこで，ストレージ側に様々な処理を能動的に行うことで，これらの管理動作の自動化を行うことを考える．その際システム側で能動的に処理を行うということを考えると，データベース上で能動的に処理を行うアクティブデータベースの考えを適用できる．アクティブデータベースではその能動的な動作を ECA ルール [5] を用いて記述しており，これを分散ストレージに対して適用していく．

データ毎にさらに様々な管理動作を行うために，HSM ではそのデータ管理のポリシーをシステムに記述しているのに対し，本研究データ毎に管理動作を記述するということを考える．現在，個々のデータの様々な情報を表すものとして，メタデータがある．メタデータは画像や動画データなどに付与し，検索などに使われていることが多い．このデータに付与するメタデータの一部として，そのデータの管理動作を記述する．管理動作を記述する際に ECA ルールを用いることにより，ストレージにて能動的な動作を行えるようにする．ECA ルールをストレージ側で読み取り，効率的に制御することにより様々な管理の自動化を達成する．これらの管理の自動化を行うための様々な提案を、Rule based Fine grain Information Lifecycle Management(RFILM) と呼ぶことにする．

今までメタデータへ操作を記述するというものがなかったのは記述するコストの大きさと，その記述をシステム側で発火するコストが非常に大きいためであると考えられる．本稿では実際にメタデータに処理を記述した場合のその動きや，その処理が現実的なコストでできることを示す．

実現方法としては，データが保存されるストレージが高機能ストレージであることを前提とする．メタデータに前述の管理動作を書き，それを高機能ストレージで実行する．高機能ストレージとして自律ディスクを採用した．

2. コンテンツへの処理記述

様々な管理動作をストレージ側で能動的に行うために，システムに管理動作を記述するのではなく，各データに管理動作を記述することを考える．その際にデータに付与されているメタデータを利用する．メタデータに ECA ルールで管理動作を記述し，ストレージ側でその動作を読み取ることによって能動的な動作を実現する．本研究では，個々のデータのメタデータに ECA ルールにて処理を記述し，ストレージ側でその処理を実現する提案を Rule based Fine grain Information Lifecycle Management(RFILM) と呼ぶ．

本節では，RFILM を実現する際のメタデータへの ECA ルールによる管理動作の記述方法と，実現における課題を示す．

2.1 ECA ルールを用いた処理記述

処理はメタデータに ECA ルールで記述を行う．前述の通り，ECA ルールは Event・Condition・Action の 3 種類からなるルールであり，これら 3 種類を記述することで大抵の動作を表現することができる．しかし，ここで問題は記述の抽象度についてである．メタデータに動作を記述するには記述する側の知識によって様々な抽象度がある．広く曖昧な表現としては，例えば「現状でもっとも空いているディスクへデータを移動する」といったように受け手によって解釈が異なる可能性がある記述に対し，細かい表現としては,「B ディスクへ当該データを移動する」といったように受け手が全員まったく同じ解釈をするような記述が考えられる．このように記述する側の知識によって，表現が異なることが考えられる．

しかしここで重要なのは，記述する側というのが必ずしも人間でなくてもよいという点である．例えば，あるアプリケーションで作成されたデータにはそのアプリケーションが自動的にメタデータに管理動作を記述を行うなど，必要に応じてアプリケーションが自動に記述してもよい．また，先ほど述べたような曖昧な表現や細かい表現を一定の抽象度にコンバートしてくれるようなコンパイラがあると想定することも可能である．本稿では全ての記述の表現を網羅することは難しいために，これらを踏まえた上で，メタデータへの記述の抽象度は記述する側がシステムが分散されたストレージで構成されていることを

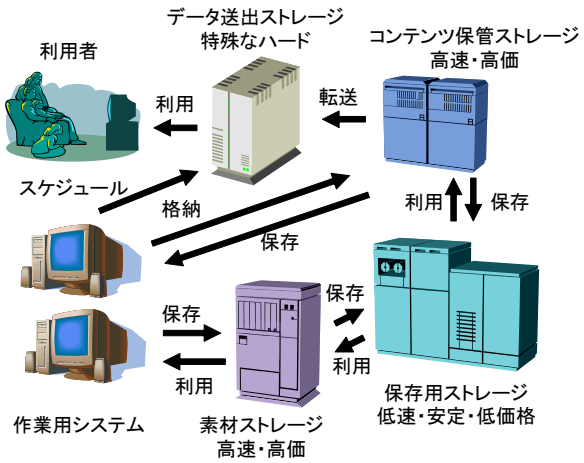

図 1　ストレージシステムの一例

知る程度であるという前提とする．この抽象度の例としては，高速でディスク容量が少ないストレージクラスタと低速で安定したストレージクラスタでシステムが構成されているなどであり，ディスクが何台でどういう配置をされているかはわからない．

2.2　ECA ルールによる記述例

ECA ルールをベースとし，以下のように Event・Condition・Action をメタデータに記述する．Action に属する部分は，高機能ストレージのルールでできることなら書くことが可能である．システム構成の一例として図 1 のシステム構成を利用し，その処理の記述例をいくつか示す．

次は送信を行ったコンテンツの削除を行うルールである．例えば，データ送出ストレージにおいて一度配送を行ったデータを削除したりであるとか，素材ストレージにおいて著作権が存在するデータがあり，その期間が切れたら削除しなければならないデータが含まれている場合などに利用できる．

Event　2005 年 1 月 28 日の 10 時になった時
Condition　このデータの送信を一度でも行っていた場合
Action　削除を行う

次はデータのアクセス制御を行うルールである．データ送出ストレージにおいて，著作権の有効期限が切れて素材として使えるデータになった場合に素材ストレージへ移動するであるとか，素材ストレージにおいて有効期限が切れたデータをその契約更新のために一時的にアクセス制御を行うといった場合に利用できる．

Event　2005 年 2 月 28 日の 15 時になった時
Condition　データの有効期限が切れていた場合
Action　一切のアクセスを許さない

次はデータの保存を行うルールである．データ送出ストレージにおいて実際に送出が行われたが再送予定がないため，次に予定が入るまではコンテンツ保管ストレージに移動しておくといった場合に利用できる．

Event　2005 年 3 月 2 日の 18 時になった時
Condition　再放送予定がない場合
Action　コンテンツ保管ストレージへ移動する

次はデータで使用頻度が高い間はストレージに保持を行うルールである．使用頻度が高いデータの場合，その使用頻度が高い間はストレージにあって欲しいことが想定される．一定間隔をルールで設定することにより，ある程度アクセスが収まったところで何かしらのアクションを起こすことが可能となる．素材ストレージにおいて，利用者が増えるような素材があった場合にはそれを多く利用する．といった場合に利用できる．

Event　2005 年 3 月 18 日の 21 時になった時
Condition　アクセス頻度が 10 以上だった場合
Action　次のアクセス頻度のチェックを 2005 年 3 月 19 日の 21 時にする

送出予定のデータの移動を行うルールである．コンテンツ保管ストレージにおいて送出時刻が近づいてきた場合，データ送出ストレージにその該当データを自動的に転送する，といった場合に利用できる．

Event　2005 年 3 月 24 日の 23 時になった時
Condition　データの送出予定が一時間以内の場合
Action　データ送出ストレージにデータの移動を行う

以上のように，メタデータに記述し，HSM では実現できないような様々な管理動作を記述することが可能である．

3.　ILM 処理の発火制御

個々のデータに付与したメタデータを全て走査し，そこに記述された ECA ルールをチェックしていたのでは時間がかかりすぎるといった問題がある [6]．また，分散環境においては発火制御情報を一元管理するようなサーバを立てると，そこがボトルネックになると考えられる．HSM においては動作管理は一元管理のサーバに記述されているが，RFILM とは異なり個別のデータの情報を全て管理するわけではないので，一元管理可能であると考えられる．

そこで，メタデータに記述された管理動作の発火を効率よく制御する必要がある．

3.1　発火制御情報の管理

HSM での一元管理が RFILM においてはボトルネックになることを考え，ネットワーク上で分散している各ノードにおいて制御情報の管理を行うことを考える．その際，発火制御情報を管理するために専用のテーブルを持つ方法と，既存のデータのアクセスパス管理のためのディレクトリを利用し，複数の管理用の機構を設けない方法がある．

本節では発火制御管理手法として，一元管理 1 手法と分散管理の 2 手法との合計 3 手法について検証を行った．

3.2　発火制御サーバを用いた集中管理

図 2 に示すように，すべてのノードとは独立に発火制御用のサーバを持つ．サーバはネットワーク越しにストレージクラスタと接続されている．そのサーバではすべてのストレージクラ

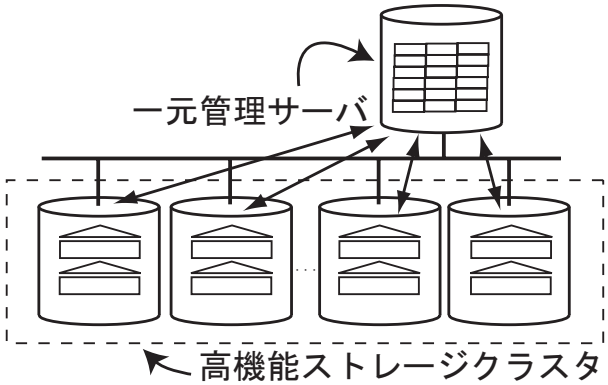

図 2　集 中 管 理

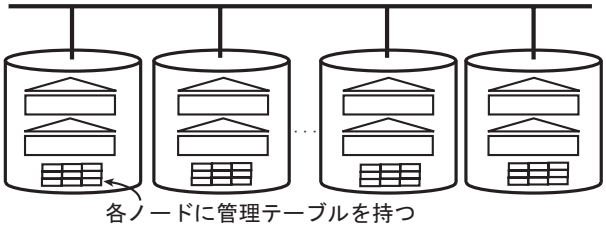

図 3　分散配置された管理テーブル

スタのノードが持つデータの発火情報の一元管理を行う．

一元管理サーバでは発火情報管理用にテーブルをもっている．管理動作の発火に関して，一元管理のノードでは管理動作の発火条件のみの情報を保持しその発火だけを制御することで，実際の管理動作はデータが含まれてるストレージのノードに任せることが可能である．つまり，テーブルには発火時刻とデータ ID の組だけを格納する．発火時刻はあるデータのメタデータに記述された管理動作の発火時刻である．また，データ ID とはそのデータを一意に特定できるような情報を表す．テーブルの内部では発火時刻順にソートを行うことで，発火時刻をチェックする際のコストを下げ，効率的な管理動作の発火を行う．

3.3　分散配置された発火管理テーブルを用いた分散管理

図 3 に示すように，各分散ノードに対して，発火制御管理用のテーブルを持たせる．テーブルには一元管理の手法と同様に発火時刻とデータ ID の組だけを格納する．テーブルの内部では発火時刻順にソートを行うことで，発火時刻をチェックする際のコストを下げ，効率的な管理動作の発火を行う．

3.4　分散 Tree のインデックスを利用した分散管理

図 4 に示すように，分散ストレージのアクセスパス管理のための分散ディレクトリに情報を付与することによって，アクセスパス管理と発火制御管理を 1 つの木によって管理を行う [6]．各インデックスノードにおいて，発火時刻とその発火時刻を持つデータへのポインタの情報を付与する．実際のデータへの最短パスを各ノードへ付与することで，発火すべきデータの特定とアクセスの高速化を行うことができる．

4.　評 価 実 験

本節では本稿における手法の有効性を確認するために 2 種類の実験を行った．

実験の 1 つ目として，説明を行った 3 種類の発火制御手法

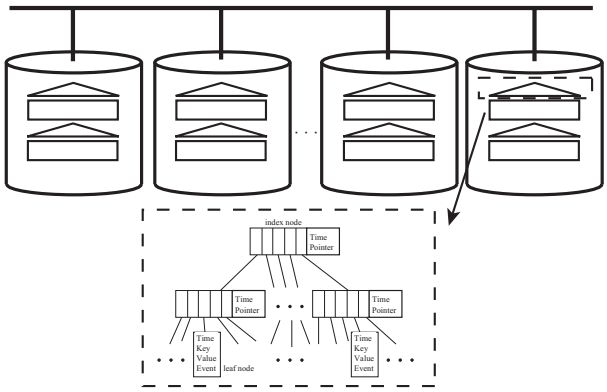

図 4　分散 Tree 上へ情報を付与

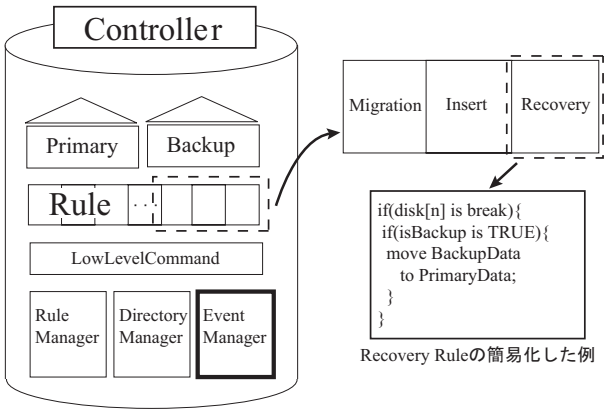

図 5　システム図

に関しての性能の比較を行った．これによって，どの手法を RFILM に適用すべきかの考察を行った．

実験のつ目として，RFILM を実現した際のコストの増加量の検証を行った．これによって，RFILM を実際に適用した場合に有用であることを示す．

本稿では RFILM の実装は高機能ストレージである自律ディスクに行った．特に今回はマルチメディアコンテンツサーバとしての自律ディスク [7] を利用し実装を行った．

詳しい実験内容や実装方法に関しては各節で説明を行う．

4.1　RFILM のシステムへの実装

RFILM を実現するにはストレージ側でメタデータに記述された ECA ルールを実行できなければならない．本節では実装の必要な条件の説明をまず行い，その条件を満たす高機能ストレージとして自律ディスクを採用しその実装法の詳細を示す．

4.1.1　実装の必要条件

高機能ストレージとして図 5 に示すように機能を有するとよい．高機能ストレージ自身がそれらの内部での処理を行うような命令郡を持ち，それらの組み合わせで様々な処理が行うことが可能ならば，ECA ルールによって様々な管理動作の実現が可能となる．高機能ストレージ内にある管理動作ハンドリングの部分を用いることで，メタデータ部分に記述された動作を行う．

4.1.2　自律ディスク

高機能ディスクである自律ディスクの概要とその特性について説明を行う．

a) 自律ディスクの概要について

コンピューターを通して扱うデータ量は増大する一方である．またそれらのデータは分散されて管理されることが多い．このデータをどのように共有し，効率良く管理するかということが重要になってきている．また，管理性の問題だけでなくストレージシステムに格納されたデータの信頼性を保たなければならない．従来の情報システム構成はどのようなサーバ機能をネットワークに接続し，機能させるかということが中心に考えられてきた．しかし我々がコンピューター上で扱うデータ量の増加に伴い，様々なストレージ装置を効率的に利用したいという要求や，情報の活用や保存をさらに高速に安全に行いたいという要求がでてきた．これらの，要求を満たすためにストレージセントリックな構成が考案された．ストレージセントリックな構成ではシステムの中心にアーキテクチャ依存のないストレージを配置する．ストレージがネットワークに直接接続されているためストレージの一元管理が可能となる．

ネットワーク接続ディスクの発展系として，近年のディスク装置中の制御用プロセッサの性能向上，キャッシュ用メモリの大容量化に伴い，ディスク装置中の機能を拡張して，ホスト側のプロセッサ上で行われていた処理をディスク上で実行される高機能化ディスクの研究がなされている．提案されている高機能化ディスクでは，従来ホストで行われていたアプリケーションをいかにディスクで行なうかということに注目されており，信頼性やディレクトリ構造，ルールを用いた高機能化，さらにはディスク自体の自律性には触れられていない．しかしネットワークに接続するディスクを考えた場合，その制御を集中して行うのは効率の良い方法とは言えない．それよりもディスク上のプロセッサの余剰能力をディスクの制御，すなわちディスクの自己管理にあてることで効率のよい処理が可能になる．

これらの問題を解決するために，ストレージ上の演算能力を用いてストレージ自体の管理を行なう自律ディスクを提案が提案されている．自律ディスクはネットワーク上の各ノードがアクティブに通信することによって処理を行なう．

b) 自律ディスクの特性について

ネットワーク接続ストレージの一元管理は様々な問題を含んでいる．しかし，一元管理は従来の分散管理よりも管理コストが大幅に低い点が評価されている．分散管理がコスト高になる一番の要因は人手による手動管理を強いられるためである．分散管理と自動管理の組み合わせにより，低管理コストと高性能を両立させることができる．

i) データ配置管理

ストレージクラスタ内でのデータ移動は負荷分散に繋がるため，積極的に行うべきである．ただし，データ移動後にもクライアントが目的のデータを見失わないようなアクセス制御を行う必要がある．

ii) 同時実行制御

データはしばしば複数のクライアントから同時にアクセスされる．そのため何らかの同時実行制御機構が必要となる．クライアント間で同時実行制御を行うには全てのクライアントにストレージ管理用のアプリケーションが必要となり，異種クライアントの混在する環境には適さない．また，管理コストの面からも得策ではない．

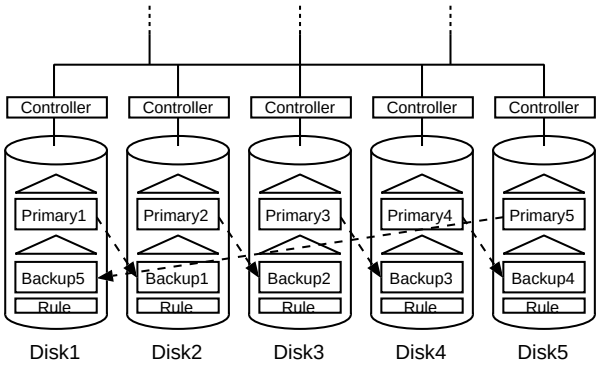

図 6　5 台の自律ディスクからなるクラスタの構成例

iii) 高い信頼性の確保

クラスタ内のストレージの故障やコントロールファームウエアのソフトウエアバグ，およびネットワークのトラブルといった障害が発生しても，クラスタ外のクライアントは障害が発生したことに気づかないほど自然に，自動的に復旧する程の高い信頼性) を確保するべきである．そのためにはクラスタ内に重複したデータを持つべきであり，その制御は当然クラスタ内で行う．

4.1.3 自律ディスクの特性

前節の考察に基づき，自律ディスクは以下の特性を持つ．

a) ネットワーク上でストレージクラスタを構成

クライアントからのアクセスをクラスタとして処理する．そのため，クライアントから見たクラスタの受け口が多く，高いスループットを有する．

b) 高機能な分散ディレクトリを採用

各自律ディスクが分散ディレクトリを持つことで，クラスタへの同時アクセスの平行処理が可能となる．そのため，スループット / レスポンスタイムの両面で有利となる．また，高機能な分散ディレクトリを採用することで，ディレクトリ自身で負荷均衡制御を行うことを考えている．

c) クラスタ構成情報

クラスタ内の通信回数を削減するために，全クラスタメンバがクラスタ構成情報をローカルに保持する．全メンバは等しい内容のクラスタ構成情報を保持する．

d) カスタマイズ可能なコマンド階層

内部コマンドとルールと呼ばれる外部コマンドを持ち，ユーザの管理戦略に合わせてルールをカスタマイズ可能とする．これは多岐にわたるユーザの要求と，記述の抽象度のバランスをとった結果である．自律ディスクで用いるルールは，アクティブデータベースで用いられている ECA ルールをベースにしたものである．ユーザはサイトの管理戦略に従いルールを簡単にカスタマイズできる．

これらの特性に基づいた自律ディスクの構成例を図 6 に示す．スループットを稼ぐためにディスクはそれぞれがプライマリ領域とバックアップ領域を保持することにする．これらの領域の配置は図 6 に示すスタガード配置に従うとする．

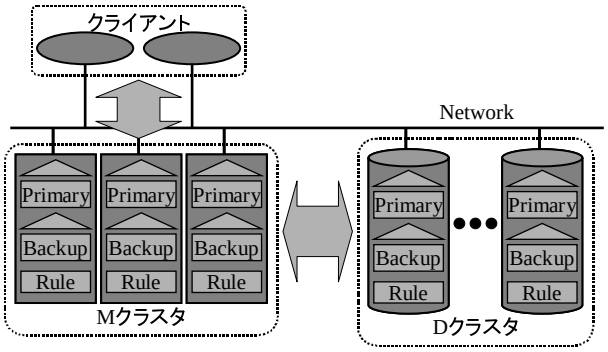

図 7　階 層 構 成

4.1.4　自律ディスクの構成例

自律ディスクの構成例としてマルチメディアコンテンツサーバ [7] や，自律ディスクの階層化を行っている．

a)　マルチメディアコンテンツサーバ

このマルチメディアコンテンツサーバにおいて，コンテンツは動画や音声などのメディアファイルや，サムネイル画像，タイトルなどのアノテーション情報から構成されている．このマルチメディアコンテンツサーバの大きな特徴は，これらのコンテンツに含まれる各オブジェクトを個々に管理するのではなく，コンテンツ単位で管理を行うことである．また，インターフェースには HTTP インターフェースを採用している．

b)　階層化を行った自律ディスク

自律ディスクでは前述の通り性能の異なるデバイスを組み合わせて階層的な自律ストレージクラスタを構成できる．この構成例を図 7 に示す．D クラスタは低速クラスタを表し，M クラスタは高速クラスタを表す．このような構成を用い，ルールを適切に変更を行えば簡単にクラスタ間の移動を行うことが可能である [8].

4.1.5　自律ディスクへの実装

模擬実装されているマルチメディアコンテンツサーバとしての自律ディスクに対し，管理動作管理モジュールを追加することで実装を行った．この章での自律ディスクとはマルチメディアコンテンツサーバとしての自律ディスクを指すものとする．ただし，発火制御サーバを用いた集中管理においては管理動作管理モジュールだけでは実装ができないので，発火制御サーバ自体を自律ディスクとは別に構築しネットワークに接続している．

a)　管理動作管理モジュールの概要

管理動作管理モジュールは，

- 管理動作の発火時刻制御部
- 管理動作の発火制御管理部

から成り，メタデータに記述された管理動作に対して管理動作の発火処理を行う．既存の自律ディスクの各ルールに対し，この管理動作管理モジュールの発火時刻制御部を実行するような実装を行った．

b)　管理動作の発火時刻制御部

発火制御サーバを用いた集中管理における管理動作モジュールに関して，自律ディスククラスタとは独立にサーバが存在している．その為，自律ディスククラスタへのデータの出入りがあったときにのみ管理動作の発火時刻の登録が起こる．つまり自律ディスクの基本的な機能では，データの挿入や削除時などに登録が起こることとなる．また，管理動作の登録はネットワークを介するために，確実に登録が行われることを保証する必要があるためにプロトコルは TCP を用いた．

他の 2 手法については，各自律ディスククラスタの個別のノードにデータの出入りが起こったときに管理動作の発火時刻の登録が起こる．発火制御サーバを用いた集中管理の場合とは異なり，ノード間のデータ移動などに登録が行われる．

それぞれの手法について，登録が発生するルールの中で管理動作管理モジュールを呼び出す動作を追加した．

c)　発火制御管理部

発火制御サーバを用いた集中管理の場合のみ，自律ディスクとは独立に制御管理部を追加した．

発火制御管理部は時刻を確認し，定期的に発火すべき管理動作がないかのチェックを行う．どの手法を用いても，時刻のチェックは登録されている一番新しい管理動作と現在時刻のチェックである．起動のチェックを行い，発火すべきであれば各手法にのっとって管理動作を発火しようとする．

表 1　実 験 環 境

CPU	Intel Pentium 3 933MHz
メモリ	PC133 SDRAM 32MB
ディスク	Seagate Barracuda IV 20.4GB 7200rpm
OS	Linux 2.2.17 glibc 2.1.3
Network	1000BASE-SX, Switching Hub
Java	Sun JDK 1.4.1 Server VM

4.2　実 験 環 境

性能測定に用いた実験環境を表 1 に示す．自律ディスクは 1 構成例であるマルチメディアコンテンツサーバとしての自律ディスクを用いた．

4.3　発火制御手法の比較

ここではに示す 3 種類の発火制御手法についての比較を行った．コンテンツの数が増えるに従い、負荷の高い管理方法ではこの管理部分がボトルネックとなって，システム全体の性能の低下を招く恐れがある．そこで，3 種類の手法の比較を行うことにより効率的な発火制御手法を示す．

4.3.1　比 較 実 験

挿入動作における管理動作の登録部分にかかる時間と管理動作を実際に発火するまでの時間の比較を行った．各ノードに 200 個と 400 個の場合を実測し，その平均値の比較を行った．この操作により各ノードには，200 個と 400 個ずつデータが入ってる場合の比較となり，クラスタ全体としては 800 個と 1600 個ずつである．また，挿入時には必ず管理動作の登録操作が行われるため，最も発火情報の登録コストの差がでやすいと考えた．

現在自律ディスクの実装上，コンテンツの挿入には自律ディ

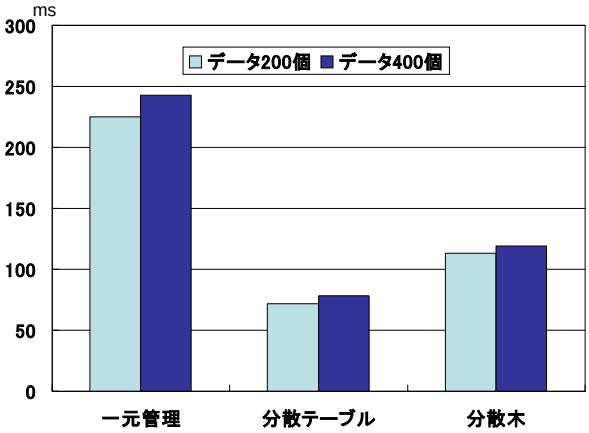

図 8　挿入のレスポンスタイム

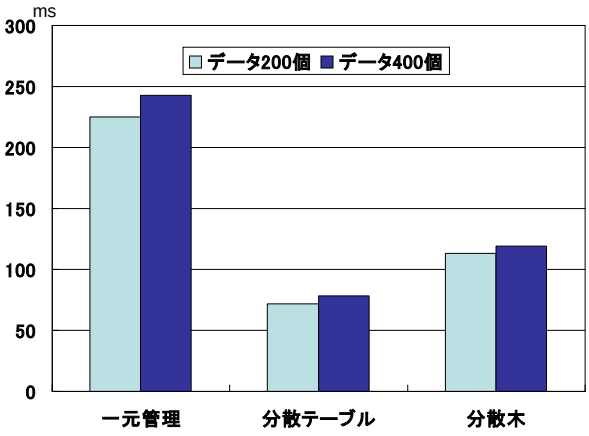

図 9　発火のレスポンスタイム

スク的な挿入を 10 回行う．ここでは管理動作の登録に関して影響がある挿入の時間の計測を行った．これはコンテンツの挿入時に起こる自律ディスク的な挿入 10 回の内の 1 回である．

4.3.2　実 験 結 果

それぞれの手法での挿入のレスポンスタイムを図 8 に示す．分散テーブルを利用した手法が最もレスポンスタイムがよかった．一元管理を行う手法は，ネットワークのオーバーヘッドが非常に大きいことがわかった．また，分散木を用いた方法にてデータを更新するコストに比べて別テーブルを更新するコストの方が小さかった．

次に，発火のレスポンスタイムを図 9 に示す．分散木のインデックスを利用した分散管理が最もレスポンスタイムがよかった．ネットワークの利用は，挿入同様オーバーヘッドが大きいことがわかった．レスポンスタイムがよかった 2 種類の差に関して，発火データの検索コストにおいては，分散木を用いた方法が最短パスで発火するデータを見つけることが可能なため，小さいと考えられる．一方で，発火後の情報の更新については分散テーブルを用いた手法の方が更新個所が少ないため小さいと考えられる．これらをふまえた上でトータルとして分散木の方がコストがかからなかったため，このような結果になったと考えられる．

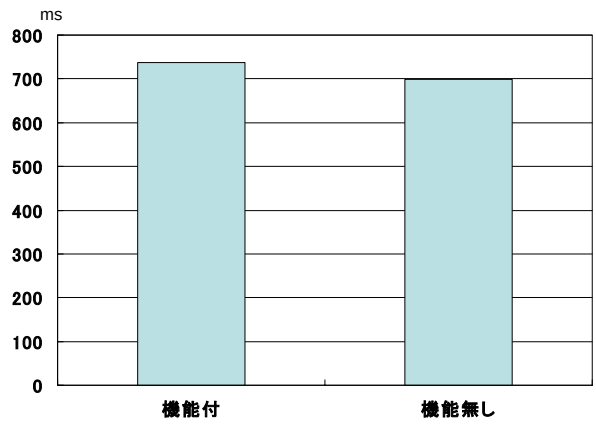

図 10　挿入のレスポンスタイム

4.3.3　考　　察

まず，一元管理の手法は最も非効率的であることが分かった．

一元管理する手法は制御情報を扱う時にネットワークを介する．他の手法は各ノード上で登録が終わるのに対して，ネットワークのオーバーヘッドが大きく影響している．また，ネットワークを利用するとネットワークのトラフィックの増大につながり，他の手法よりネットワークコストの増加も考えると，分散ノード上で発火制御を分散管理した方がよいということがわかる．さらに一元管理の場合，ノード数やコンテンツ数が増加していくと，破綻することが予想される．

挿入に関しては分散テーブルを用いた手法の方がレスポンスタイムがいい結果となったが，故障処理などにおける木の統合や，データの数に対する耐性を考えると分散木を利用した制御手法の方がいいと考える．

4.4　RFILM を実装したシステムの性能

分散木のインデックスを利用した分散管理を適用した RFILM を実装したシステムとそれらの実装がされていないシステムの比較を行った．

挿入とデータの取得に関してそのリクエストにかかる時間を比較した．200 回実測し，その平均値を記録した．

挿入に関して，現在自律ディスクの実装上，データのすべての情報を挿入するのに自律ディスク的な挿入を 10 回行っている．今回の実験結果はこの 10 回の挿入を一回のコンテンツの挿入として実験結果としているため，値が大きくなっている．

4.4.1　実 験 結 果

図 10 に挿入のレスポンスタイムを，図 11 にデータの取り出しのレスポンスタイムを示す．FILM を導入することによって，挿入に関して約 11.8 %，データの取得に関して約 5.6 %の性能の低下を計測することができた．

4.4.2　考　　察

全く新しい機能を加えたにも関わらず，オーバーヘッドはわずかであった．

例えば，記述例で述べたような削除を行うという操作を考えた場合，本稿の機能を有していない場合は多大なコストがかかる．まず一点目にどのコンテンツを削除するべきかということを管理する必要があるということである．本手法では人間側の

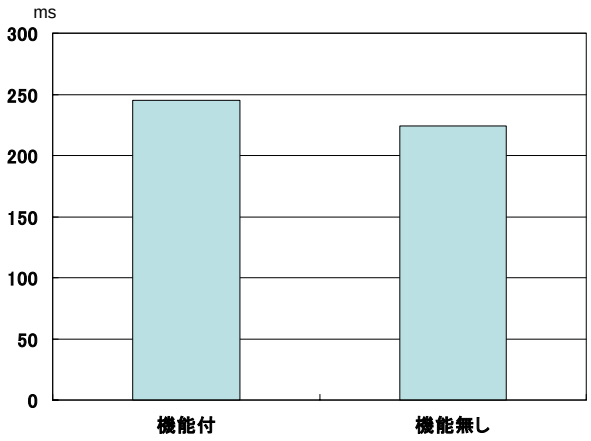

図 11　データの取得のレスポンスタイム

管理コストは 0 になる．二点目は手順を人間側が知る必要があるということである．削除する手段やその方法はストレージによって異なり，人間側はそれを学ぶ必要性がある．三点目にミスがなくなる，ということである．人間は間違いを犯してしまうことが多々あるが，ストレージ側が勘違いすることはないためである．

RFILM を実装をわずかなオーバーヘッドのみで実現し，データ管理における人的コストの削減を達成した．

5. 結　　論

5.1 ま と め

データ量の増加と，データの種類や利用方法の多様化によって，データの管理するコストが高くなってきているため，効率よく管理することが要求されている．本研究では，データの管理するコストの削減を行うために，ストレージ側で能動的に管理動作を行うことを考えた．さらに，HSM では達成できない様々な管理動作を行うためにデータに付与したメタデータに ECA ルールを用いて記述し，それをストレージ側で実行することによりストレージ側で能動的な動作を行うことを提案した．これを RFILM と名づけた．

RFILM を行う上で，管理動作の発火制御が必要であることを実験により示した．発火制御が必要であることから，HSM で利用されている一元管理を行う手法と，分散している各ノードにてそれぞれ管理する手法を考えた．分散している各ノードでの管理において，発火情報用の管理テーブルを持つ手法と，データのアクセスパス管理の木を利用した手法の 2 種類を考え，一元管理の手法とあわせ 3 種類に対して考察を行った．

実験により，発火制御を分散している各ノードにて行った方がよいことを示した．さらに，分散している各ノードにおいては，クラスタ内部で起こる様々な動作を考えると，分散木のインデックスを利用した手法がよいことを示した．

また，RFILM を実装したシステムと RFILM を持たないシステムの比較を行った．その結果として 11.8 %の性能の低下のみで，RFILM の実現ができた．システムに新しい機能を加えたにも関わらず全体的な性能を落とさず，システムの管理者の負荷を軽減できることを示した．

5.2　今後の課題

ストレージシステムの例で示したような，細かい動作を指定したいシステムを実際に構築してみるべきである．今回はデータの数が実際使われているストレージより少ないと考えられるので，さらに多くのデータ数で実験してみる必要がある．また，管理動作の記述に関して，記述すること自体のコストや記述する抽象度に関して今回は一定での抽象度でという仮定で行った．この部分で検討が必要である．今回はデータにメタデータを付与する方法をとったが，MXF などのように 1 つのバイナリの中にデータとメタデータが一緒になっているものもある．データとメタデータに分けた場合は二種類のデータの管理を行わなければならないが，1 つのバイナリであればその必要はない．しかし，データの読み出しやメタデータの読み出しを行う際にそれらのデータを取り出す必要が生じてしまう．この点に関してさらなる検討が必要である．

6. 謝　　辞

本研究の一部は，科学技術振興事業団戦略的創造研究推進事業 CREST，情報ストレージ研究推進機構（SRC），文部科学省科学研究費補助金特定領域研究（16016232）および東京工業大学 21 世紀 COE プログラム「大規模知的資源の体系化と活用基盤構築」の助成により行われた．

文　　献

[1] F. Golshani: “Multimedia Information Lifecycle Management”, Multimedia,IEEE, **11**, (2004).
[2] K. H. Bernd Reiner: “Optimized Management of Large-Scale Data Sets Stored on Tertiary Storage Systems”, IEEE DISTRIBUTED SYSTEMS ONLINE 1541-4922, **5**, (2004).
[3] O. G. Harry Hulen: “Storage Area Networks and the High Performance Storage System”, 10th NASA Goddard Conference on Mass Storage Systems and Technologies in cooperation with the 19th IEEE Symposium on Mass Storage Systems (2002).
[4] D. Corporation: “enigma”.
[5] e. a. Dayal, U.: “Combining Active Databases and Timing Constraints”, ACM SIGMOD Record, pp. 51–70 (1988).
[6] 山口, 渡邊, 小林, 田口, 林, 上原, 横田：“分散ストレージにおける情報ライフサイクルの効率的管理”, 信学技報，TECHNICAL REPORT OF IEICE. DE2004-78(2004-07) 電子情報通信学会 (2004).
[7] 渡邊, 花井, 山口, 横田：“自律ディスクを用いたマルチメディアコンテンツサーバ”, No. DE2002-86, DC2002-22 (2002).
[8] 花井, 渡邊, 山口, 田口, 林, 上原, 横田：“半導体ディスクを用いた自律ディスクの階層化”, 情処学会研究会報告, データベースシステム DBS-131-19 情報処理学会 (2003).