半導体ディスクによる自律ディスククラスタの階層化構成

Hierarchical Architecture of Autonomous Storage with Solid State Disks

花井 知広♡ 渡邊 明嗣♡ 山口 宗慶♡
田口 亮♢ 林 直人♢ 上原 年博♢
横田 治夫♠

Tomohiro HANAI Akitsugu WATANABE
Munenori YAMAGUCHI Ryo TAGUCHI
Naoto HAYASHI Toshihiro UEHARA
Haruo YOKOTA

我々は，ストレージシステムにおいて負荷分散，故障対策，障害回復などの高度な機能を実現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．しかし，ストレージシステムに対する性能要求はより高まってきており，磁気ディスクに対して半導体メモリを組み合わせることが重要になってきている．本稿では，磁気ディスクによる自律ディスククラスタと半導体ディスクによる自律ディスククラスタから構成される，自律ストレージの構成手法を提案する．それらのクラスタ間での耐故障キャッシュメカニズムを実現するために，3 種類の挿入・更新プロトコルを提案する．さらにこれらのプロトコルと磁気ディスクのみで構成される自律ディスククラスタとの比較実験を行い，我々の手法の有効性を示す．

We have proposed autonomous disks to import advanced functionality into storage systems, which balance loads, tolerate faults, and recover data within a cluster. However, since the higher performance is strongly required for the storage systems, a combination of magnetic disk with semiconductor memory is significant. In this paper, we propose a hierarchical architecture of autonomous storage constructed from a magnetic autonomous disk cluster and a solid state autonomous disk cluster. To realize fault tolerant cache mechanisms between these clusters, we propose three types of protocols. Using an experimental system, we compare these protocols with a cluster constructed from only magnetic autonomous disks to demonstrate the effectiveness of our approach.

1. はじめに

大規模データ処理システムにおけるストレージ管理コストの増加に伴い，ストレージをシステムの中心に据えるストレージセントリックシステムが注目されている．我々はストレージシステムにおける負荷分散，故障対策，障害回復などの高度な機能を実現するためのアプローチとして，ディスク装置内の制御用プロセッサとキャッシュ用メモリを利用する自律ディスクを提案してきた [1].

♡ 学生会員　東京工業大学 大学院 情報理工学研究科 計算工学専攻
{hanai,aki,muu}@de.cs.titech.ac.jp

♢ NHK 放送技術研究所
{taguchi.r-cs,hayashi.n-gm,uehara.t-jy}@nhk.or.jp

♠ 正会員　東京工業大学 学術国際情報センター
yokota@cs.titech.ac.jp

しかし，ストレージシステムで扱われるデータ量は未だ増加の一途をたどり，その性能に対する要求もより高いものが求められている．例えば典型的なトランザクションシステムの処理性能向上のために，より高速なレスポンスがストレージに対して求められている．しかし，現在のストレージシステムの基礎となる記憶媒体は磁気ディスクであり，その構造から飛躍的な速度向上は望めない状況である．

そこで本稿では磁気ディスクを用いて構成された自律ディスククラスタ上に，半導体ディスクを用いてキャッシュ用の自律ディスククラスタを構成することによる，自律ディスクを用いたストレージシステムの高速化を検討する．また，アベイラビリティを保ちつつ効率的な挿入・更新処理を行うプロトコル，効率的なキャッシュ管理アルゴリズムについても検討を行う．その上で提案手法を試作システム上に実装し，性能測定により提案手法を評価する．

2. 自律ディスクの概要

まず自律ディスクの概要について説明する．自律ディスクはネットワーク環境でディスククラスタを構成することを前提としている．クライアントはデータにアクセスするためにクラスタ内の任意のノードに要求を発行し，クラスタ内のノードはノード間の局所的な通信を行うことで，協力してクライアントからの要求に対処する．標準的な構成ではクラスタ内の各ノードは，プライマリディレクトリと他ノードのバックアップを行うバックアップディレクトリを持つ．データに対するアクセスは分散ディレクトリをトラバースし，リクエストを適切なノードに転送することにより行われる．このような前提のもとで，自律ディスクはデータ分散，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，異種性といった性質を持つ．自律ディスクの詳細については文献 [1] を参照されたい．

3. 自律ディスクの階層化

磁気ディスクより高速なメディアを用いたストレージデバイスとして半導体ディスクが存在する．しかし半導体ディスクはアクセス速度が高速であるが，容量あたりの単価が非常に高く，ストレージシステム全てを半導体ディスクで構成するのは現実的でない．そこで現在の一般的なストレージシステムでは，半導体ディスクを磁気ディスクを用いたストレージシステムのキャッシュとして用いている [2, 3]．この手法は自律ディスクに対しても有効であると考えられるので，磁気ディスクを用いた自律ディスククラスタに対して，半導体ディスクを用いてキャッシュシステムを構成することを考える．そのような階層化ストレージを構成する上で，ストレージシステム全体として既存の自律ディスクと同じ機能を提供する必要がある．つまり，ノードの追加・削除がオンラインで可能であり，管理コストを増加させてはならない．また，半導体ディスクは揮発性メモリの利用も想定するものとする．つまり，電源断などの障害が発生した場合には半導体ディスク内のデータが失われてしまう可能性がある．このため半導体ディスク内のデータは冗長化されなければならない．

このような条件のもとで，磁気ディスクのみを用いて構成された自律ディスククラスタと，半導体ディスクのみを用いて構成された自律ディスククラスタを考える．以下，磁気ディスクのみを用いて構成されたクラスタを D クラスタ (Disk-Cluster)，半導体ディスクのみを用いて構成されたクラスタを M クラスタ (Memory-Custer) と呼ぶことにする．D クラスタと M クラスタは同一のネットワーク上に存在し，M クラスタがクライアントからの要求を受ける．D クラスタは直接クライアントとの通信は行わない．このアーキテクチャの概要を図 1 に示す．D クラスタ，M クラスタともに同一の自律ディスクソフトウェアを用い，異なるのはハードウェア構成とルールのみである．このような構成にすることにより，D クラスタや M クラスタの構成変更をそれぞれ独立に行うことができる．また，M クラスタではキャッシュの配置管理が自律ディスクの分散ディレクトリによって行われる．このためキャッシュが保持

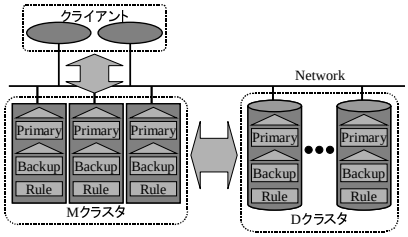

図 1: 階層化アーキテクチャ
Fig.1 Hierarchical architecture

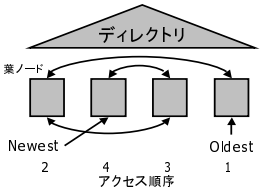

図 2: 葉ノードによる LRU リストの構成
Fig.2 LRU list of leaf nodes

されるべきノードが一意に決まり，分散キャッシュシステムで問題になるキャッシュ一貫性問題を避けることができる．

3.1 キャッシュ管理アルゴリズム

キャッシュシステムの構成には，キャッシュ容量に対してエントリが溢れた際に，どのエントリをキャッシュから追い出すかを決定するための置換アルゴリズムが重要である．エントリへのアクセス履歴記録や追い出しエントリを決定する際のオーバーヘッドが小さいものが必要である．また追い出しエントリとしては，頻繁にアクセスされるエントリではなく，アクセス頻度が低いエントリを選ぶべきである．現在のマイクロプロセッサ内部では LRU(Least Recently Used) アルゴリズムがキャッシュ制御アルゴリズムとして広く用いられている．一方，自律ディスクでは分散ディレクトリとして Fat-Btree などの分散 Btree を利用する．そこで本稿では Btree の葉ノードにおいて効率的に LRU アルゴリズムを利用する手法を提案する[1]．

M クラスタ内で追い出しエントリを選ぶ際には，M クラスタ全体の中で最も使われていないエントリを選ぶのがキャッシュヒット率の観点からは望ましい．しかしそれでは追い出し要求が発生する度に M クラスタ全体に問い合わせを行う必要があり，スケーラビリティが問題となる．そこでクラスタの各ノード内でのみ LRU アルゴリズムを行い，追い出し要求に対処するものとする．ノード間にキャッシュ利用率の偏りが発生した場合は，自律ディスクの持つ負荷均衡化機構により偏りを除去する．葉ノードは図 2 のように最近アクセスされた順に LRU リストと呼ばれる双方向リンクリストを構築する．Btree へ新たに葉ノードが挿入されると，LRU リストへは先頭に挿入される．Btree から葉ノードが削除された場合には，LRU リストからも削除される．読み出し，更新などの葉ノードへのその他のアクセスの場合は，その葉ノードはいったん LRU リストから削除され，LRU リストの先頭に挿入される．これらの操作はいずれも一定時間で行うことができるため，キャッシュエントリの管理コストを非常に小さく抑えることができる．

3.2 挿入・更新プロトコル

次に，自律ディスクの階層化構成における挿入・更新プロトコルを提案する．ここで，階層化構成における挿入・更新プロトコルは以下の条件を満たす必要がある．まず単一故障に耐えるために，データを 2ヶ所以上に冗長化して保持する必要がある．また，半導体ディスクを揮発性であると仮定しているので，データを永続化するためには必ず磁気ディスクへ書き込まなければならない．以上の条件に従って，3 つの挿入・更新プロトコルを提案する[2]．

3.2.1 Write-Through-Sync(WTS) プロトコル

この手法では，M クラスタはプライマリディレクトリのみを持ち，バックアップディレクトリを持たない．挿入・更新時には M クラスタのプライマリ，D クラスタのプライマリとバックアップに同期的に書き込み，クライアントに完了通知を返す．このプロトコルにおける処理の流れを図 3 に示す．

1. クライアントが M クラスタの任意のノードに対して Insert リクエストを送る．
2. M クラスタのプライマリディレクトリにデータを書き込む．その後 M クラスタは D クラスタに同内容の Insert リクエストを送る．
3. D クラスタはプライマリとバックアップディレクトリに同期的に挿入・更新処理を行い，M クラスタへ完了通知を返す．
4. M クラスタはクライアントへ完了通知を返す．

キャッシュエントリの追い出し時には，単に追い出しエントリをプライマリから削除するだけでよい．このプロトコルは最も単純な手法であり実現は容易であるが，3 箇所に同期書き込みを行うためリクエスト処理にかかる時間が大きくなる．

3.2.2 Write-Through-Async(WTA) プロトコル

この手法でも，M クラスタはプライマリディレクトリのみを持ち，バックアップディレクトリを持たない．挿入・更新時には M クラスタのプライマリと D クラスタのプライマリへ同期的に書き込みを行い，クライアントに完了通知を返した後に，D クラスタのバックアップを非同期に生成する．このプロトコルにおける処理の流れを図 4 に示す．

1. クライアントが M クラスタの任意のノードに対して Insert リクエストを送る．
2. M クラスタのプライマリディレクトリにデータを書き込む．その後 M クラスタは D クラスタに同内容の Insert リクエストを送る．
3. D クラスタはプライマリディレクトリへ挿入・更新処理を行い，M クラスタへ完了通知を返す．
4. M クラスタはクライアントへ完了通知を返す．
5. その後 D クラスタは任意のタイミングでバックアップディレクトリへ挿入・更新処理を行い，M クラスタへバックアップ生成完了通知を送る．
6. M クラスタはバックアップ生成完了通知を受け取ると，対応する葉ノードにマークをつける．

最後のステップ 6 はキャッシュ追い出し時に D クラスタ内のバックアップの有無を確認するためのものである。マークされていないならば D クラスタにバックアップを生成してから追い出す必要がある．この手法では WTS プロトコルに比べ，D クラスタでのバックアップ生成がクライアントのリクエストとは非同期に行われるため，レスポンスタイムが短縮される．

3.2.3 Delayed-Write(DW) プロトコル

この手法では，M クラスタはプライマリとバックアップディレクトリを持つ．挿入・更新時には M クラスタのプライマリとバックアップに同期書き込みを行い，クライアントに完了通知を返す流れである．このプロトコルにおける処理の流れを図 5 に示す．

1. クライアントが M クラスタの任意のノードに対して Insert リクエストを送る．
2. M クラスタのプライマリとバックアップディレクトリに同期的にデータを書き込み，クライアントへ完了通知を返す．
3. M クラスタは任意のタイミングで D クラスタに同内容の Insert リクエストを送る．
4. D クラスタはプライマリとバックアップディレクトリで同期的に挿入・更新処理を行い，M クラスタへ完了通知を返す．
5. M クラスタは完了通知を受け取ると，対応する葉ノードにマークをつけた後，バックアップディレクトリからエントリを削除する．

最後のステップ 5 が必要な理由は WTA プロトコルの場合と同様である．また，バックアップの削除は記憶領域の有効利用のため

[1] ここでは葉ノード 1 つの容量がインデックスノード 1 つの容量よりも大きい場合を考え，インデックスノードの使用容量は無視できるとし，葉ノードの使用容量のみを考慮する．

[2] 自律ディスクはログを用いた非同期バックアップによりプライマリとバックアップの同期更新を避け挿入・更新処理コストを抑える手法も提案されているが，ここでは簡単のため同期更新を行う場合について説明する．

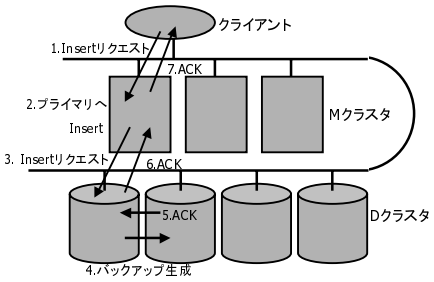

図 3: Write-Through-Sync プロトコル
Fig.3 Write-Through-Sync protocol

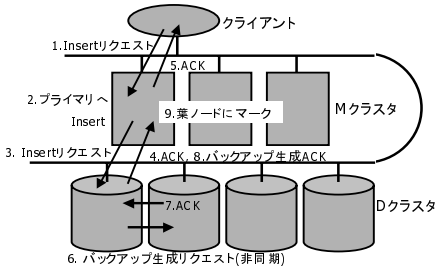

図 4: Write-Through-Async プロトコル
Fig.4 Write-Through-Async protocol

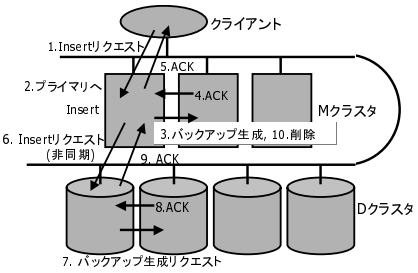

図 5: Delayed-Write プロトコル
Fig.5 Delayed-Write protocol

表 1: 実験システムの構成
Table 1 The experimental system

	D クラスタ	M クラスタ
ノード数	6 台	3 台
CPU	Intel Pentium 3 933MHz	Intel Xeon 2.4GHz, HT
メモリ	PC133 SDRAM 256MB	PC2100 DDR SDRAM 1024MB
ディスク	Seagate Barracuda IV 20.4GB 7200rpm	BiTMICRO E-Disk 4GB FC-HBA:QLogic QLA2310
OS	Linux 2.2.17	Linux 2.4.20 SMP
Network	1000BASE-SX, Switching Hub	
Java 環境	Sun JDK 1.4.1	

である．この手法では挿入・更新時のプライマリとバックアップをともに M クラスタ内に生成することにより，レスポンスタイムを短縮する．

3.3 読み出し時の動作

読み出し時には以下のプロトコルで処理を行う．

1. クライアントが M クラスタの任意のノードに対して Retrieve リクエストを送る．
2. M クラスタのプライマリディレクトリをトラバースする．リクエストされたデータが M クラスタ内に存在するならば，それをクライアントへ返し処理終了．存在しない場合は D クラスタへリクエストを転送する．
3. D クラスタはプライマリディレクトリからリクエストされたデータを読み出して M クラスタへ返す．
4. M クラスタは受信したデータをクライアントへ返すとともに，プライマリディレクトリに挿入する．

4. 性能評価

ここでは前章で述べた階層化手法の実験による評価を行う．まず前述の挿入・更新プロトコルとキャッシュ置換アルゴリズムを，我々が現在 PC 上に Java を用いて実装を行っている試作システム上に実装を行った．その上でリクエスト処理にかかる時間を計測し，階層化を行わない場合，階層化を行った場合の各プロトコルの評価を行う．

4.4 測定手法

性能測定に用いた実験環境を表 1 に示す．半導体ディスクは Fibre Channel で接続されたフラッシュメモリを用いた．またクライアントには D クラスタノードと同じ構成の PC を 1 台用いる．

この環境に対して，以下の手法で性能測定を行う．Insert，Retrieve リクエストをそれぞれ一定回数発行し，処理完了までの所要時間を計測する．リクエストは 6000 個のキー集合を用い，連続して 30000 回行う．格納されるデータのキーは 16 バイトのランダムな文字列であり，データサイズは 1K バイトとする．キーの偏りによる性能の変化を評価するために，各キーの利用頻度を Zipf 分布とし，偏りの度合いを表すパラメータ θ を 0.0〜1.0 に変化させて測定を行う．

表 2: M, D クラスタの性能測定
Table 2 Performance of M and D clusters

	Insert	Retrieve
M クラスタ	123370 [ms]	52190 [ms]
D クラスタ	181513 [ms]	80681 [ms]

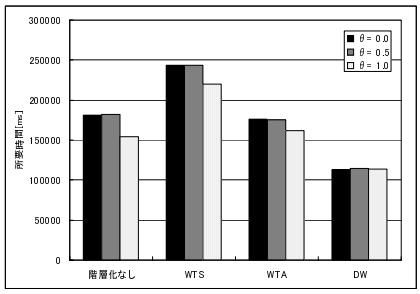

図 6: Insert 操作の実行時間
Fig.6 Execution time for Insert operations

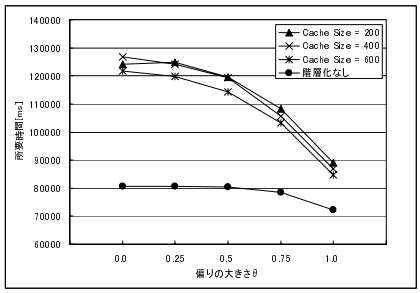

図 7: アクセス偏りと Retrieve 操作の実行時間
Fig.7 Access skew and performance for Retrieve operations

4.5 予備実験

各操作の性能測定を行う前に，M クラスタと D クラスタのそれぞれの性能を測定する．それぞれのクラスタで独立に従来の自律ディスククラスタを構成し，アクセス偏りのない状態で Insert, Retrieve 操作の性能を計測した結果が表 2 である．この結果より，この実験システムにおいて M クラスタは D クラスタに比べ，Insert で 68%, Retrieve で 65%程度の時間で同等のリクエストを処理できることがわかる．

4.6 挿入・更新処理性能の測定

前述の手法によって Insert 操作の性能を測定した結果を図 6 に示す．測定の際には M クラスタにおける最大キャッシュエントリ数を 200 としている．

DW プロトコルを用いた場合は階層化を行わない構成に比べ，偏りがない場合で 38%程度の時間短縮となった．これは DW プロトコルがプライマリとバックアップをともに M クラスタ内に Insert することにより，Insert リクエストのレスポンスタイムが短縮されたことがわかる．一方，WTS プロトコルでは 3 つのノードに書き込みを行ってからクライアントへレスポンスを返すため，階層

化しない場合に比べレスポンスタイムが悪化していることがわかる．WTA プロトコルではプライマリを M クラスタに書き込むため，理論上は階層化なしの場合に比べレスポンスタイムが短縮されるが，D クラスタ内での非同期バックアップ生成のコストや，M クラスタにおけるキャッシュ管理のオーバーヘッドによって，階層化なしの場合と同程度の速度になっている．

Insert リクエストにおいて各キーの発生頻度の偏りが大きい場合は，DW プロトコルの場合を除いて，速度が向上している．これは各キーの発生頻度に偏りがある場合では，各ノード上のディスクキャッシュにヒットする頻度が向上するためであると考えられる．

4.7 読み出し処理性能の測定

前述の手法により Retrieve 操作の性能を，M クラスタにおける最大キャッシュエントリ数を変化させて測定した結果を図 7 に示す．

各キーの発生頻度に偏りが大きい場合では，M クラスタがキャッシュとして効率的にはたらくため，キャッシュ容量を増やすにつれて速度が向上していることがわかる．階層化を用いない手法についても，偏りが大きくなるにつれて各ノード上のディスクキャッシュにヒットする頻度が向上するために速度が向上している．しかし，より偏りが大きい場合においては階層化手法と逆転する可能性がある．

一方，偏りがない場合では同一データが高頻度でアクセスされることがないため，キャッシュ容量が増えるに従って速度が低下してしまっている．これは，キャッシュ容量が増えるにつれて M クラスタにおけるディレクトリのトラバースコストが増加するためである．さらにキャッシュ容量を増加させるとキャッシュヒット率の向上により速度が向上している．

4.8 考察

挿入・更新処理性能については，階層化手法における DW プロトコルが階層化しない場合を大きく上回り，階層化手法の有効性を示した．

一方，読み出し処理性能についてはリクエストに偏りがない場合では階層化手法は階層化を行わない手法に大きく離され，偏りがある場合についても差は小さくなるが階層化を行わない手法の方が高速であるという測定結果になった．ここで階層化手法における読み出し処理性能が，階層化を行わない場合に比べて高速になる条件を考える．M クラスタにおけるキャッシュヒット率を r，M クラスタでの Retrieve 操作の所要時間を T_M，D クラスタでの Retrieve 操作の所要時間を T_D とすると，Retrieve 操作に対する平均所要時間は $rT_M + (1-r)(T_M + T_D) = T_M + (1-r)T_D$ である．これが T_D よりも小さくなるためには，$r > \frac{T_M}{T_D}$ である必要がある．今回実験を行ったシステムでは予備実験より $\frac{T_M}{T_D} \simeq 0.65$ であった．また $\theta = 1.0$，最大キャッシュエントリ数が 600 の時で $r \simeq 0.75$ であり，このとき階層化なしで偏りなしの場合とほぼ同程度の所要時間となっている．キャッシュ管理の通信オーバヘッドを考慮すると，これは条件をほぼ満たしているといえる．$\frac{T_M}{T_D}$ がより小さいシステム，つまりクラスタ間の速度差がより大きいシステムでは更にこの条件を達成することは容易であるので，階層化により高速な Retrieve 操作を行うことができると考えられる．

5. まとめと今後の課題

本稿では，半導体ディスクを用いた自律ディスククラスタをキャッシュとして用い，磁気ディスクを用いた自律ディスククラスタと組み合わせることによる自律ストレージの構成手法を提案した．アベイラビリティを損なわない効率的な挿入・更新プロトコルや，Btree 上で効率的に LRU キャッシュ置換アルゴリズム実現する手法を提案した．

また，それらのプロトコルやアルゴリズムを実験システム上に実装し，性能測定により有効性を確認した．実験では挿入・更新操作における Delayed-Write プロトコルは階層化を行わない場合に比べ，約 38%の処理時間短縮が達成された．読み出し操作については実験システムの構成上の理由で階層化を行わない場合よりも性能が低下したが，アクセスに偏りがある場合についてはその有効性を確認することができた．

本稿で述べた構成手法は，自律ディスクを用いた場合だけでなく，SAN 等の他のアーキテクチャに対しても適用可能であり実用性が高いと考えられる．

本稿では自律ディスククラスタの階層化アーキテクチャを示したが，クラスタ間の性能差や台数比よっては M クラスタがボトルネックになることが考えられる．どのような場合に M クラスタがボトルネックになりえるのかの理論的な考察が必要である．

今回の性能評価では比較的小さなデータを用いたが，データサイズが大きい場合にも本稿の手法は有効であり，スループットを大きく向上させることができると考えられる．自律ディスクにおいて大きなデータを効率的に扱う手法として提案している間接ディレクトリ方式 [4] と組み合わせた場合の性能評価が必要である．

[謝辞]

本研究の一部は，文部科学省科学研究費補助金特定領域研究 (15017233) 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．

[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] Imperial Technology, Inc. Tuning Storage Area Networks (SANs) with Solid State Disk. http://www.imperialtech.com/pdf/Tuning_SANs_Whitepaper-.pdf.

[3] Solid Data. SAN Cache: SSD In The SAN. http://www.soliddata.com/pdf/storageinc.pdf.

[4] 花井知広 , 横田治夫. 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. In **情処学会研究会報告**, **データベースシステム** *DBS-128-29*. 情報処理学会, 2002.

花井 知広　Tomohiro HANAI
平 14 東工大・工・情工卒．同大大学院・情報理工・計算工・修士課程在学中．日本データベース学会学生会員．

渡邊 明嗣　Akitsugu WATANABE
平 14 東工大大学院・情報理工・計算工・博士前期課程了．同大大学院・情報理工・計算工・博士後期課程在学中．日本データベース学会学生会員．

山口 宗慶　Munenori YAMAGUCHI
平 15 東工大・工・情工卒．同大大学院・情報理工・計算工・修士課程在学中．

田口 亮　Ryo TAGUCHI
平 6 慶應義塾大大学院・理工・計測工・修士課程了．同年より NHK 放送技術研究所．映像情報メディア学会会員．

林 直人　Naoto HAYASHI
昭 58 金沢大・工・精密工卒．同年より NHK 放送技術研究所．映像情報メディア学会会員．

上原 年博　Toshihiro UEHARA
昭 56 慶應義塾大・工・電気工・修士課程了．昭 59 より NHK 放送技術研究所．電子情報通信学会，映像情報メディア学会各会員．

横田 治夫　Haruo YOKOTA
昭 55 東工大・工・電物卒．昭 57 同大大学院・情報・修士課程了．同年富士通 (株)．同年 6 月 (財) 新世代コンピュータ技術開発機構研究所．昭 61(株) 富士通研究所．平 4 北陸先端大・情報・助教授．平 10 東工大・情報理工・助教授．平 13 東工大・学術国際情報センター・教授．工博．日本データベース学会, 電子情報通信学会, 情報処理学会，人工知能学会，IEEE，ACM 各会員．