計算機アーキテクチャ　112－1
（１９９５．　６．　２３）

# 並列二次記憶における動的アクセス分散による負荷の平滑化

大上靖弘　北村徹　大西一正　清水雅久

RWCP[1]超並列三洋研究室[2]

超並列計算機で実行されるアプリケーションの多くは大量のデータを使用するものであり、超並列計算機には大規模な二次記憶を有することが要求される。また、超並列計算機ではプロセッサから二次記憶に対して多数のアクセス要求が同時に発生するため、これらを多数の二次記憶装置で並列に処理できる能力も必要である。このように膨大な記憶資源を有効に利用し、アクセスの並列度を高めるためには、二次記憶装置全体の稼働率を高めることが重要な課題である。本論文では、プロセッサからの多数のアクセス要求を各二次記憶装置の負荷に応じて動的に分散し、アクセスの並列度を向上する手法DECODE(Dynamic Express Changing Of Data Entry)を提案し、ソフトシミュレーションによりアクセス性能を評価する。

# Dynamic Load Balancing of File Access on Parallel Secondary Storage

Yasuhiro OUE, Toru KITAMURA, Kazumasa OHNISHI and Masahisa SHIMIZU

Massively Parallel Systems Sanyo Laboratory, RWCP.

Mostly applications executed on the massively parallel computer treat massive amount of data. Massively parallel computer must have enormous amount of secondary storage. And also, on the massively parallel computer, lots of processors generate the number of data requests to the secondary storage simultaneously, and these data requests must be executed by massive secondary storage in parallel. So, it is important to enhance the operation rate for all disk to permit efficient use of its extensive resources by users. This paper proposed a parallel file access method named DECODE(Dynamic Express Changing Of Data Entry). This method realizes high-concurrency for disk access by balancing the number of data requests according to workload of each disk. The evaluation results by software simulation of this method shows good performance.

[1]RWCP:Real World Computing Partnership(新情報処理開発機構)
[2]三洋電機(株)東京情報通信研究所内

## 1 はじめに

並列計算機の二次記憶システムでは、複数のプロセッサからの多数のアクセス要求が同時に発生するため、高い並列度とバンド幅が要求される。このような高い性能を備える二次記憶システムとして、多数のディスクを並列に接続し、各ディスクに対して並列にアクセスを行なうディスクアレイが有効となる。また、超並列計算機が扱う問題は一般的に大規模であり、それらの問題では膨大なデータが使用される。大規模な問題では、例えば、Grand Challengeのように100GBから1TBのデータを必要とするアプリケーションもある[1]。従って、これらの並列二次記憶システムにおいては、並列計算機の規模の拡大に伴い、数百から数千台のディスクを接続した大規模並列ディスクシステムが必要となることが予想される。

このような大規模な並列ディスクシステムにおいてスケーラビリティを確保するためには、各ディスクをいかに有効に活用するかが重要となる。各ディスクを高い稼働率で動作させるためには、プロセッサからのアクセス負荷を一部のディスクに偏ることなく、各ディスクに対して均等に割り当てることが必要である。

並列ディスクに対する負荷分散の手法としては、従来から、データを各ディスクに分散配置するストライピングが用いられている。予想されるアクセスパターンに応じて、ストライプ幅や分散配置するディスク数を調整することによって、アクセス負荷を均等に割り当てることが可能である。しかし、並列計算機においては、リソースの有効利用のため、プロセッサの領域分割によって複数のアプリケーションが並行して実行される。このような実行環境においては、個々のアプリケーションのアクセスパターンに対する最適な配置は必ずしも有効ではない。また、複数のアプリケーションによるアクセスパターンを予測することは困難である。従って、ストライピングのような静的なデータ配置戦略では並列ディスクシステムの性能を十分に活用することができず、負荷の変動に対応して、動的に負荷を分散することが必要となる。

我々は、並列二次記憶システムに対する負荷に応じて動的な負荷分散を行ない、並列に接続された二次記憶装置全体の性能を引き出すことを可能とするアクセス方式DECODE(Dynamic Express Changing Of Data Entry)を提案する。本方式では、データ更新時に書き込み先のディスクを動的に選択し、負荷の低いディスクに対して選択的にWRITEアクセスを行なうことにより負荷分散を実現する。

本論文では、2章でDECODE方式について説明し、本研究で想定する並列ディスクシステムの構成と、その構成に対するDECODE方式の適用を説明する。3章では、シミュレーションによる性能評価を行ない、4章でその結果について考察を加える。5章でまとめと今後の課題を述べる。

## 2 DECODE方式による動的負荷分散

DECODE方式では、データ更新時に書き込み先のディスクをディスク負荷の状況に応じて動的に選択する。これにより、アクセス負荷のディスク間での動的な負荷分散を実現する。

以下に、本研究で想定する並列ディスクシステムの構成と、その構成に対するDECODE方式の適用について説明する。

### 2.1 並列ディスクシステム

並列計算機に複数のディスクを接続する形態として、ディスクを各プロセッサに接続する形態、I/O管理を行なう特別なプロセッサを設けてディスクを接続する形態、プロセッサ間接続網とは別に入出力専用のネットワークを設けてそこにディスクを接続する形態などが考えられる。我々は、どのプロセッサからも全てのディスク上のデータが一様にアクセスできること、命令実行系と入出力系の性質の異なる通信を効率良く行なうことができることから、入出力専用のネットワークを介してディスクを接続する形態が望ましいと考える[2]。

また、負荷分散のためにはディスクの負荷状況を把握して、アクセス負荷の投入先を決定しなければならないが、これを行なう方式として、集中管理方式と分散管理方式がある。集中管理方式では、単一のディスパッチャが全ディスクに関する情報を収集し、アクセス負荷の投入先を決定する。情報を一元管理でき、構成が容易であるが、プロセッサからのアクセス要求が単一のディスパッチャに集中するため、プロセッサ側の並列度が上がり多数のアクセス要求が発生するとボトルネックになり易い。分散管理方式は複数のディスパッチャがディスクに関する情報を重複して管理する。プロセッサからのアクセス要求を分散できるので並列度が上がるが、ディスクに関する情報の一貫性の維持や最新情報の獲得のためのオーバーヘッドが必要で、制御も複雑になる。我々は、ディスクを階層構造

とし、集中管理と分散管理の組合せによって制御する。具体的には、比較的少数のディスクによるディスククラスタを集中管理し、さらにディスククラスタ間で協調して分散管理を行なう。

従って、我々の想定する並列ディスクシステムは図1のような構成となる[3]。

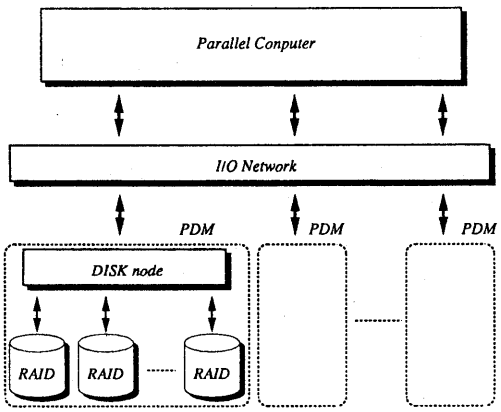


図 1: 並列ディスクシステム

ディスクは、PDM(Parallel Disk Module)と呼ばれるディスククラスタ単位でI/O専用のネットワークに接続される。ディスクとしては信頼性の向上のためにRAID技術を採り入れたディスクアレイを用い、PDMは8台程度のRAIDディスクと1台のディスクノードによって構成される。ディスクノードは負荷分散のための情報収集とアクセス負荷の投入先の決定、キャッシュの管理、RAIDディスクの制御を行なう。

### 2.2 PDM間の負荷分散

PDM間の負荷分散は、ディスクノードが負荷情報を分散管理し、アクセス負荷の転送によって実現する。基本的には、プロセッサはデータ更新要求をそのデータの存在するRAIDディスクを制御するディスクノードに対して行なう。ディスクノードは他PDMの負荷状況を把握しておき、自分よりも負荷の低いPDMに対してデータ更新要求を転送する。この時に問題になるのは、他PDMの負荷状況の把握である。

全てのPDMの状況を把握するためには、PDM間で負荷情報のブロードキャストが必要となる。負荷情報の即時性という意味では頻繁なブロードキャストが望ましいが、オーバーヘッドとのトレードオフとなる。また、全PDMが同じ情報を持つと、最も負荷の低いPDMに対して多数のPDMから負荷が転送されるという事態も起こり得る。そこで、PDMをグループ化して、各グループ内で負荷情報の収集とアクセス負荷の転送を実行する。グループ化することによって負荷情報のブロードキャストによるオーバヘッドを抑えることができる。

一方、アクセス負荷の転送をグループ内に制限することで、負荷分散がシステム全体に対して行なわれず、グループ間での負荷の偏りが発生することが考えられる。この問題は、グループを互いに重複させ、1つのPDMが複数のグループに含まれるように構成することによって解決する。(図2)

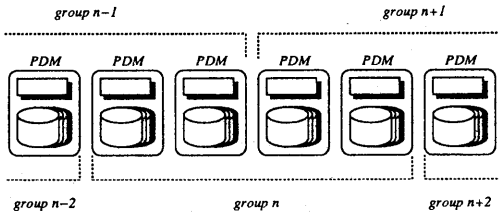


図 2: PDMのグループ化

さらに、負荷情報をグループ内のPDMに対してブロードキャストする際には、負荷の変動を逐一伝えるのではなく、負荷に対する段階的な閾値を用意して、負荷のレベルが変化した時にのみブロードキャストすることによって、オーバーヘッドの削減を行なう。

### 2.3 PDM内の負荷分散

PDM内の負荷分散は、RAIDディスクの負荷状況をディスクノードが集中管理し、PDM内の負荷が均等になるようにRAIDディスクを選択してWRITE操作を行なうことによって実現する。

ここで、データ更新時のディスク負荷状況によって、ファイルデータの配置が変化していくことによるファイルのフラグメンテーションが問題になる。特に、ランダムな書き込みの後に、同じファイルに対してシーケンシャルな読み出しを行なう場合、フラグメンテーションが性能劣化をまねくと考えられる。

従って、データ更新の際には、フラグメンテーションを抑えるように、ディスク内の書き込み位置を選択する必要がある。本システムでは、データ更新によって発生する古い無効なデータが占める領域を選択的に再利用することによってフラグメンテーションを抑える。これらの再利用可能な領域は、ディスクノードがファイル毎に各ディスクのフリー領域リストによって

管理する。

フラグメンテーションの抑制のためには、同一のファイルは連続した領域に書かれることが望ましい。また、ファイルの隣接する論理ブロックは、隣接する物理ブロックに存在することが望ましい。一方、負荷分散のためには、負荷の最も低いRAIDディスクに対して負荷を投入することが望ましい。

従って、ディスクノードは、負荷の分散とフラグメンテーションの抑制の両方を考慮して、次のようにRAIDディスクへの負荷の投入を行なう。

まず、WRITE操作によるデータの書き込み先の領域を次の3種類に分類する。(図3)

**Original Position(OP)**

そのデータの元の位置。フラグメンテーションは起こらない。

**Equivalent Position(EP)**

同じファイルによって占められている領域で、ファイルの論理的な順序とディスク上の物理的な順序が一致する位置、及びその周辺領域。周辺領域のサイズは、ファイル毎に設定される値によって決定される。フラグメンテーションの影響が少ない。

**New Position(NP)**

WRITE操作にあたって、新しく獲得される領域。フラグメンテーションの影響が大きい。

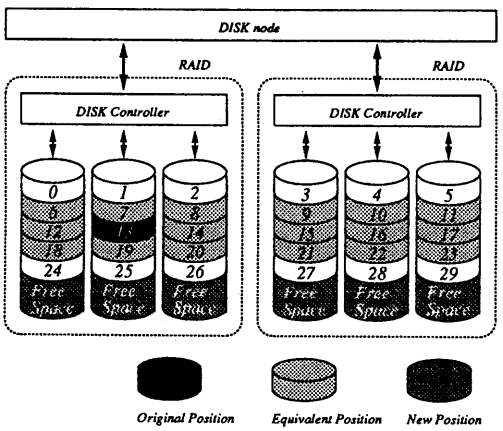


図 3: データ書き込み領域の分類

負荷に大きな偏りがなければ、ディスクノードはOPにデータを書き戻す。しかし、偏りが生じる場合は、負荷の高いRAIDディスクから、負荷の低いRAIDディスクに対して書き込みを移動する。

まず、負荷の高いRAIDディスクにOPがあるデータに関して、負荷の最も低いRAIDディスクから順にEPに空き領域があるかどうかを調べる。空き領域があれば、そのデータの書き込みは移動される。EPに空き領域があるデータを全て移しても、負荷に大きな偏りがある場合は、さらに、負荷の高いRAIDディスクにOPがあるデータを負荷の最も低いRAIDディスクのMPに移す。

また、ファイルデータの物理的な配置がデータ更新によって変化していくため、データ管理構造(inode)もその都度変更する必要がある。この影響で、データ更新のたびにinodeを更新することによるオーバーヘッドが発生するが、inodeをデータと連続した位置に同時に書き込むことによって、inode更新のオーバーヘッドを隠蔽する。

## 3 性能評価

並列ファイルアクセス方式DECODEによる負荷分散の有効性を検証するための第1段階として、PDM内の負荷分散の効果についてシミュレーションによる評価を行なった。

アクセスを行なうファイルのサイズは10MBとし、ファイル生成時はデータが4KBのサイズで各ディスクにストライプされているものとする。シミュレーションでは、このファイルに対してデータ位置をストライプサイズ単位で動的に変えるDECODE方式と、データ位置を変えない従来の方式の2種類の方式によってアクセスを実行し、平均アクセス応答時間の比較を行なう。

なお、各アプリケーションが異なるファイルにアクセスする場合を想定した。

### 3.1 シミュレーションモデル

シミュレータは、機能レベルでの記述が可能なシミュレータツールSES/workbenchを用いて作成した。今回の性能評価においては、PDM内の負荷分散による効果の検証が目的であるため、ディスク8台で構成される1つのPDMにおける動作のシミュレーションを行なった。以下に想定したモデルの構成をまとめる。

- ディスク8台が並列に接続され、1つのディスクノードによって制御される。

- ディスクにはRAID-3のディスクアレイを用いる予定であるが、モデルの簡易化のために、シミュレーションでは通常の単一ディスクを用いる。ディスクのパラメータを表1に示す。
- ディスクノードは、ディスクに対して発行したアクセス要求のサイズから算出したアクセス時間を各ディスクの負荷として使用する。

表 1: ディスクのパラメータ

| | |
|---|---|
| 平均シーク時間 | 12 [ms] |
| 回転速度 | 5400 [rpm] |
| セクタサイズ | 512 [B] |
| セクタ数 / トラック | 100 |
| トラック数 / シリンダ | 25 |

### 3.2 実験条件

シミュレーションに用いるアプリケーションのアクセスパターンとしては以下の2種類を使用した。この2種類のアクセスパターンのいくつかの組合せでシミュレーションを行なう。

**ランダムアクセス**

ファイル全体に対してランダムにアクセスを行なう。

アクセスサイズ 3種

512B ～ 8KB(1 ～ 16 セクタ、ランダム)

512B ～ 16KB(1 ～ 32 セクタ、ランダム)

512B ～ 32KB(1 ～ 64 セクタ、ランダム)

アクセス頻度 200 ～ 800KB/s

READ/WRITE 比 3:1

**シーケンシャルアクセス**

ファイルの先頭からシーケンシャルにアクセスを行なう。

アクセスサイズ 1MB

アクセス頻度 10MB/s

READ/WRITE 比 1:1

### 3.3 実験内容

実験では、以下の3つのケースについてシミュレーションによりアクセス性能を評価した。

(a) ランダムなアクセスパターンを有する2つのアプリケーションを同時に実行する。

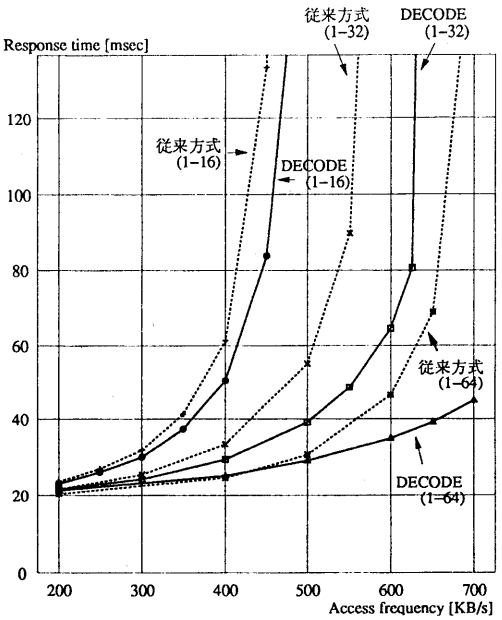


図 4: ランダムアクセス (a)
アクセス頻度に対する WRITE の応答時間

(b) シーケンシャルなアクセスパターンを有する2つのアプリケーションを同時に実行する。

(c) ランダムなアクセスパターンを有するアプリケーションとシーケンシャルなアクセスパターンを有するアプリケーションを同時に実行する。

### 3.4 実験結果

図4、図5はランダムなアクセスが生じるアプリケーション二つを同時に実行した時 (a) の応答時間を示したものである。横軸は1つのアプリケーションが単位時間あたりに発生するアクセスの頻度を表しており、縦軸が平均の応答時間を表している。

WRITE アクセスに関しては、DECODE 方式では負荷分散の効果によって性能向上が達成されており、例えばアクセスサイズ 1-64 セクタの場合、アクセス頻度が 650KB/s の時に約 43% の性能向上が達成されている。また、従来方式の応答時間はアクセス頻度が高くなるにつれて急激に増大する。これに対して DECODE 方式ではアクセス頻度が高くなると負荷分散の効果が大きくなるため性能が改善され、応答時間の増大は緩やかである。

READ アクセスに関しては、アクセス頻度が低い

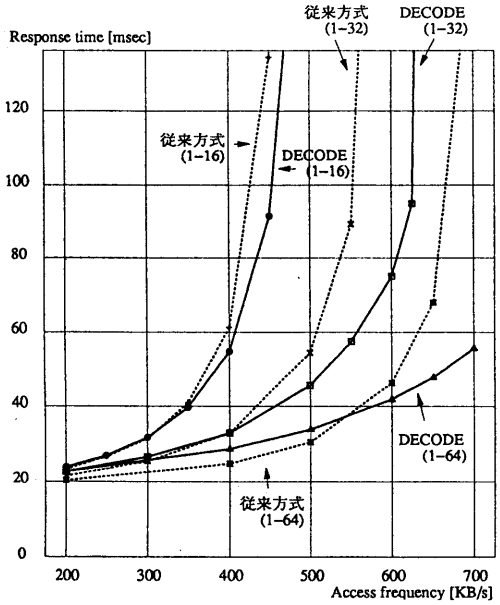


図 5: ランダムアクセス (a)
アクセス頻度に対する READ の応答時間

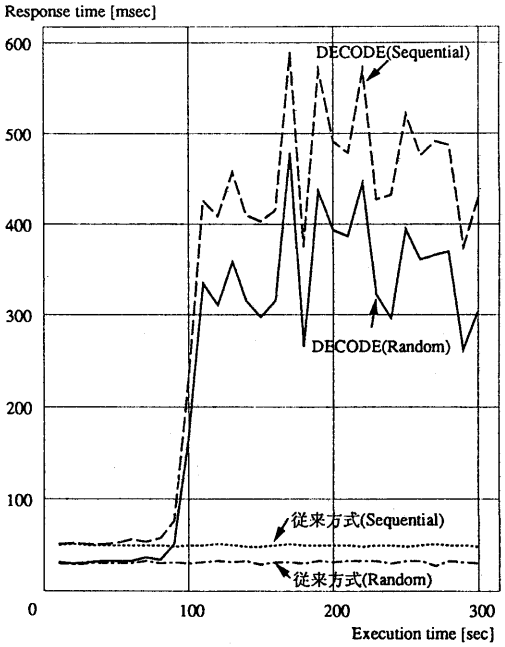


図 6: ランダム & シーケンシャル (c)
実行時間に対する READ の応答時間

領域では WRITE アクセスによる負荷分散の効果に対してフラグメンテーションの影響が大きいため、DECODE 方式は従来方式に比べて 2%-16% 程度、応答時間が増加している。しかし、アクセス頻度が高くなるに連れてフラグメンテーションの影響よりも負荷分散の効果が大きくなり、DECODE 方式は従来方式の性能を上回っている。例えばアクセスサイズ 1-64 セクタの場合、アクセス頻度が 650KB/s の時に約 30% の性能向上が達成されている。また、応答時間の増大は、WRITE アクセスの場合と同様に緩やかになっている。

シーケンシャルなアクセスが生じるアプリケーション二つを同時に実行した場合 (b) は、負荷の不均衡が生じないため、データ位置の移動が起こらず、DECODE 方式と従来方式のアクセスは全く同じ応答時間となる。

図 6はシーケンシャルなアクセスが生じるアプリケーション (10MB/s) と、ランダムなアクセスが生じるアプリケーション (200KB/s) を同時に実行した時 (c) の応答時間を READ アクセスについて示したものである。この図では横軸が実行時間であるが、DECODE 方式では実行が進むに連れシーケンシャルアクセスの性能がフラグメンテーションによって大幅に劣化しており、その影響でランダムアクセスの性能も劣化している。

フラグメンテーションの影響は WRITE アクセスの移動が行なわれたデータサイズとアクセスサイズの関係によって決まる。今回の実験では、シーケンシャルなアクセスが 1MB 単位で行なわれるのに対して、書き込みの移動が 4KB 単位である。このように、負荷分散がアクセスサイズに対して小さな単位で行なわれるため、フラグメンテーションの影響が大きくなり、性能が劣化していると考えられる。

そこで、シーケンシャルなアクセスが行なわれるファイルのストライプサイズを 128KB とし、書き込みの移動を 128KB で行なうことによって負荷分散を比較的大きな単位で行なった。ここで、従来方式においては、ストライプサイズを 4KB とした場合と 128KB とした場合では、アクセスサイズが 1MB であれば全く同じ応答時間を示す。この時の応答時間を、図 7、図 8に示す。横軸はランダムなアクセスを生じるアプリケーションが単位時間あたりに発生するアクセスの頻度を表しており、縦軸が平均の応答時間を表している。ここでは、シーケンシャルなアクセスを生じるアプリケーションのアクセスの頻度は 10MB/s としてい

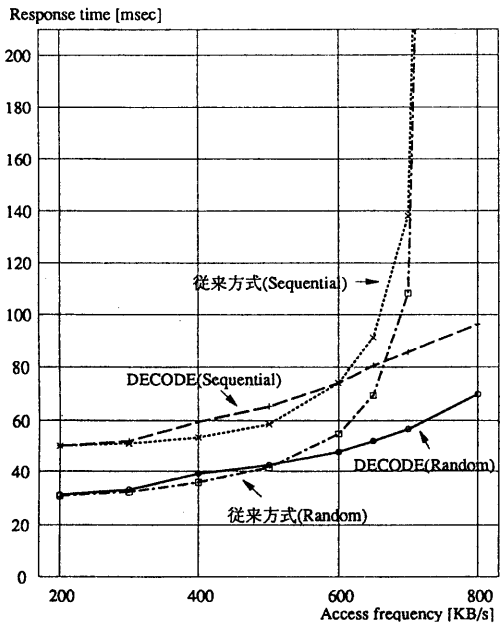


図 7: ランダム & シーケンシャル (c) 改善後
アクセス頻度に対する WRITE の応答時間

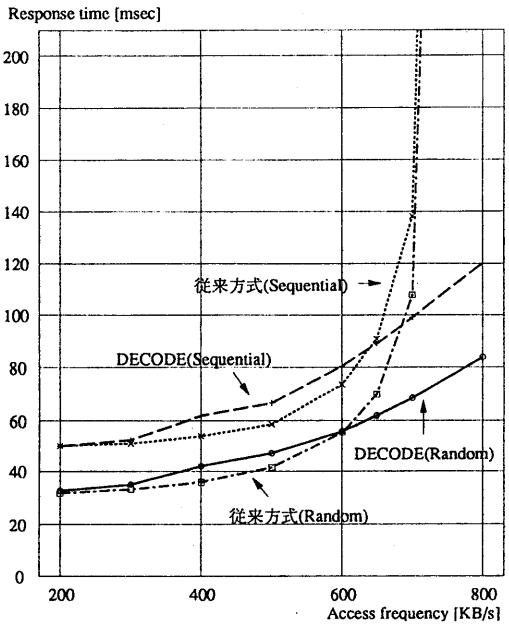


図 8: ランダム & シーケンシャル (c) 改善後
アクセス頻度に対する READ の応答時間

る。

この図では、図 4、図 5と同様に性能向上が達成されており、例えばアクセス頻度が 700KB/s の時、ランダムアクセスが生じるアプリケーションについては、WRITE アクセスで約 48%、READ アクセスで約 37% 性能が改善されている。また、シーケンシャルアクセスが生じるアプリケーションについても、WRITE アクセスで約 38%、READ アクセスで約 28% 性能が改善されている。アクセス頻度に対する DECODE 方式の応答時間の増大も、従来方式に比べて緩やかになっている。

## 4 考察

### 4.1 アクセス頻度と性能の関係

DECODE 方式の性能は、フラグメンテーションの影響と負荷分散の効果によって決定される。フラグメンテーションには WRITE アクセスの移動が行なわれたデータサイズとアクセスサイズの関係が影響を及ぼし、負荷分散の効果は個々のディスクのアクセス負荷量に対するディスク間の負荷の偏りの割合によって決まる。

アクセス頻度が低い場合には、ディスクの負荷が低くディスク間の負荷の偏りも小さいために、負荷分散の効果も小さい。従って、フラグメンテーションの影響によって負荷分散の効果が打ち消され、従来方式と同等の性能もしくは若干下回る性能となる。

アクセス頻度が高い場合には、ディスク間の負荷に大きな偏りが発生するようになるので、従来方式では頻度がある値を越えたところで応答時間が急激に増大している。これに対して、DECODE 方式では負荷分散の効果が大きくなる。一方で WRITE アクセスの移動が多くなるため、フラグメンテーションも発生するが、2.3節で述べたように、データ更新によって発生する古い無効なデータが占める領域を選択的に再利用することによってフラグメンテーションを抑えることができるので、負荷分散の効果がフラグメンテーションの影響を上回るようになり、結果として応答時間が改善される。

### 4.2 アクセスパターンによる影響

シミュレーション評価の結果が示すように本方式の効果はアクセスパターンに影響を受ける。当然、アクセス負荷が各ディスクに対して均等に生じるアプリケーションについては、効果はほとんどない。しかし、ランダムなアクセスについては、大局的には二次記憶全体に対して負荷が均等に発生するが、短時間に

ついて考えるとアクセス負荷が不均等であるため、大幅な性能向上が達成されている。従って、アクセスパターンが複雑で、データの最適配置が困難な場合には、大局的にもアクセス負荷が不均等になり、本方式の効果がさらに顕著になると予想される。

### 4.3 WRITEの移動サイズ

WRITEの移動を小さな単位で行なえば、負荷分散をより細かい粒度で実行できる。しかし、アクセスサイズに比べてWRITEの移動サイズが小さ過ぎると、フラグメンテーションによる影響が大きくなる。また、負荷分散の効果は、ディスク間の負荷の偏りと各ディスクの負荷量によって決まり、負荷量に対して偏りが小さければ、負荷分散の効果は小さくなる。つまり、負荷分散を細かい単位で行なうと効果があるとは限らない。

従って、WRITEの移動を行なう単位は、アクセスのサイズ、負荷の状況等によって決定することが必要である。

## 5 まとめ

本論文では、二次記憶へのアクセス負荷を平滑化することにより、並列計算機の膨大な記憶資源を効率的に利用する手法を提案した。本手法は、データを書き込むディスクを各ディスクの負荷状況に応じて動的に決定するものであり、プロセッサからの多様なアクセスに対し、アクセス負荷に応じた動的な負荷分散を実現する。

次に、本研究で想定する並列ディスクシステムの構成を述べ、DECODE方式の具体的な実現手法として、ディスククラスタ(PDM)間とPDM内の2段階の負荷分散を提案した。その際問題となるPDM間の負荷情報の通信とファイルのフラグメンテーションの対策として、各々PDMのグループ化とフラグメンテーションの抑制を考慮したデータ位置の選択について述べた。最後に、PDM内の負荷分散に関してソフトシミュレーションによる性能評価を行ない、アクセス負荷を平滑化することにより大幅なアクセス性能の向上が達成されることを示した。

今後は、より多くのアクセスパターンについて性能評価を行なうとともに、ディスククラスタ間の負荷分散の効果に関してシミュレーションによる検証を行なう。さらに、超並列計算機上に本方式を構築し、実機上で有効性の検証を行なっていく予定である。

## 参考文献


[1] J.M.del Rosario, A.N.Choudhary. High-Performance I/O for Massively Parallel Computers. IEEE Computer March 1994, (1994), 59-68.

[2] 大西、北村、大上、清水. 超並列計算機における並列二次記憶の基本アーキテクチャ. 情報処理学会第48回全国大会論文集, 3B-5, (1994).

[3] 大上、北村、大西、清水. プロセス間のアクセス競合を低減する並列二次記憶システムの構想. 情報処理学会研究報告 ARC-106-7, (1994), 49-55.