修 士 論 文

冗長ディスクアレイ用
耐故障分散ディスクキャッシュに関する研究

指導教官　横田 治夫 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

小島 信

1997 年 2 月 14 日

目 次

第 1 章

序論

1.1 本研究の背景と目的

近年、計算機システムが適用される分野は拡大の一途をたどり、日常生活を含む様々な処理にコンピュータが適用されるようになった。システムの規模は年を追う毎に増大し、マルチメディアを筆頭にした最近の大規模システムは、コンピュータシステムに対して、より高い信頼性、処理の高速性、大容量アクセスなどを要求する。

システムの高速化を実現する一方の要素である、CPU やメモリ等のマイクロデバイスは、アーキテクチャやプロセス技術の向上によって、年に 40 %から 60 %性能向上を実現している。しかし、もう一方の要素である二次記憶装置は、現在主流の磁気ディスクを用いたものにおいては、機械的動作を伴うことなどから、密度こそ年に 60 %から 80 %の割合いで大容量化しているとはいえ、その性能は過去 10 年では 1 年につき 7 %から 10 %程度の向上にとどまっている。そのため現状では、CPU がいかに高速になろうとも周辺二次記憶装置の低速な入出力によってシステム全体の性能は低く抑えられてしまう。従って、これらの問題を解決する事がコンピュータシステムの発展には不可欠であると考えられている。

これらの問題に対応すべく登場したのが冗長ディスクアレイ、あるいは RAID (Redundant Arrays of Inexpensive Disks) と呼ばれる外部記憶システムである。これらは、複数のハードディスク装置を並列に動作させることでシステムのアクセス処理速度を高速化し、データに冗長情報を付加することで信頼性を向上させる [2]。

RAID はデータ及び冗長情報の格納方法によって幾つかのレベルに分類されており、

このうち RAID レベル 3、4、5 と呼ばれるものはパリティ情報を利用した高いエラー耐故障機構を実現している。これらにおいては、ディスクアレイ中の１台のディスクが故障した場合でも別のディスク上に保持しておいた冗長データを用いる事によって故障したディスクのデータを復元する事が可能である。

しかし、RAID レベル 4、5 は書き込みが遅いと言う欠点が存在する。それはディスクへの書き込みの度に新しいパリティ情報を算出しなければならないため、データ自体の書き込みとは別に付随的なデータのアクセスを行なわなければならないからである。このアクセス速度の低さを改善するため、RAID に大容量の write-in キャッシュを搭載する事が一般的である。しかし、これらキャッシュ上のデータに対するデータ保護は特になされていない。耐故障性を持った RAID 用ディスクキャッシュの研究としては Jai Menon が 2 重化したディスクキャッシュ方式を発表している [1][3]。これは同じデータを 2 台のキャッシュに置く事で、片方のキャッシュに障害が起きてもデータはもう 1 つのキャッシュ上に生き残っていると言うシステムである。

本研究では同じように耐故障性を持った RAID 用ディスクキャッシュ機構として、新たに分散ディスクキャッシュシステムを提案する。キャッシュを用いる事で高速な書き込みを実現しながら、各々独立したキャッシュメモリを RAID の各ディスクに分散、データだけでなく冗長情報もキャッシュ上に置く事で耐故障性能を持たせる。そして冗長情報としてパリティを用いる事で、2 重化ディスクキャッシュより冗長度を抑え、結果としてメモリ効率を上げる事により、より高い書き込み速度性能を持つキャッシュシステムの構築を試みる。

1.2 本論文の構成

本論文の構成は以下のようになる。

まず 2 章では、従来のディスクアレイシステムとして、RAID の説明と、本研究の比較モデルとした RAID 用ディスクキャッシュの概要を説明する。

3 章では、分散ディスクキャッシュ付き RAID システムの仕組み、通常運航時の動作内容とタイムテーブル、故障時の対応などについて概説する。

4 章では、今回データ測定に用いた実際の HD にアクセス可能な分散ディスクキャッシュ付き RAID システムの実働モデルのハードウェア構成と、ソフトウェア構造について詳述する。

実験システムを用いた性能測定や、他のシステムとの比較した場合の優位性に関する評価は、5 章で行なう。

最後の章で、本研究のまとめと今後の課題を述べる。

第 2 章

従来の RAID と RAID 用ディスクキャッシュ

二次記憶装置の性能を向上させるために、現在さまざまな試みが行なわれている。しかし、システムに対して高性能な部品を用いるとか、付加的な部品を取り付ける事による機械性能を向上させるような場合、それはそのままシステム自体のコストアップに繋がる事が多い。この章で説明するディスクアレイは、一般的なハードディスクを複数台集めて連帯させ、仮想的に一台の二次記憶装置として扱う事で、二次記憶装置の信頼性と速度面での性能向上を低コストで実現したシステムである。

2.1 RAID

ディスクアレイは一般的なディスクを複数台、互いにデータを保障しながら並列に用いる事で、群体として 1 台の二次記憶装置として総括的に見た場合の速度性能と信頼性を向上させようとするものである。

速度面に関しては、複数のディスクに対して並列アクセスや、データストライピング（分割）を行なう事で、アクセス速度を大幅に向上させる事が出来る。また、データの冗長化情報を複数のディスクへ配置する事によって高い信頼性をシステムに与える事が可能である。

ディスクアレイの種類として UCBerkeley の D.A.Paterson らによって提案されたのが RAID である。

RAID では書き込みの際にデータを複数のディスクに格納すると同時に、冗長情報も分散保持する事でシステムの信頼性と速度性能を向上させている。RAID はデータ及び冗長情報の格納方法によって、以下のようにレベル 0 からレベル 5 までの分類がなされている。

- ［レベル 0］データストライピング（分割）により、高いデータ転送レートを得る方式である。ただし、データの冗長情報は保持しないのでシステムとしての信頼性は個々の記憶装置に依存する。

- ［レベル 1］ミラーリングとも呼ばれる。全てのデータを二重化し、冗長情報として別々のディスクに保持する事で、片方に障害が生じても残った方のディスクを用いる事で処理をそのまま続行する事が出来る。シンプルで信頼性も極めて高く、また一方が故障した後の性能の劣化もないと言う優れた長所がある反面、データの冗長度が高い為、記憶効率が悪い。従来からファイルサーバなどに良く用いられてきた。

- ［レベル 2］データはビット単位で各ディスクに格納され、冗長情報としてハミングコードを用いてエラーの検出及び訂正を行なう事が出来る。データアクセス時には全てのディスクを必要とするため、大規模アクセスに適しているが、小規模のアクセスには適していない。次に示すレベル 3 と比べると記憶効率が悪いため、あまり使用されていない。

- ［レベル 3］レベル 3 では、冗長情報としてパリティを用いて、データのエラー訂正を行なっている。ディスク故障はディスクのコントローラによって検出するようにする。冗長情報が格納されているディスクはパリティグループの中の 1 台で済むため記憶効率が良い。データはレベル 2 と同様、ビットまたはバイト単位で各ディスクに格納されるため、アクセス時には全ディスクが必要となり、やはり小規模のアクセスには適さない。信頼性に関しては、一台までのディスク故障に対応する事が可能である。レベル 5 とともに現在、最も良く使用される。

- ［レベル 4］レベル 3 において、データ分割単位をセクタ単位のブロックとし、ブロックに対するアクセスをディスクごとに独立に行なうことが可能となり、性能が向上する。パリティーデータはパリティグループ内の一つのディスク保持されており、データ書き込み時にパリティを更新する際、パリティディスクにアクセスが集中するという問題がある。

図 2.1: RAID レベル 5 の構成

この場合、横一列のデータがパリティグループを形成すると想定している。パリティ情報は、グループを構成するデータの排他的論理和から求められる。

$$\text{パリティデータ} = \text{データ} \oplus \text{データ} \oplus \cdots \oplus \text{データ} \tag{2.1}$$

これから、ディスク故障のせいでグループ中のデータのどれか１つが消えた場合でも、下式のように同じグループのその他のデータを全て排他的論理和する事で、無くなった

データを復元する事が出来る。

$$\text{失われたデータ} = (\text{残り全てのデータの排他的論理和}) \oplus \text{パリティ} \tag{2.2}$$

この仕組みによって、RAID レベル 3、4、5 は強力な耐故障性を持つ。しかし、このためグループのパリティデータはグループデータに対して常に整合性を維持する必要があり、したがってデータの書き込みを行なう際はグループのパリティデータも同時に算出して更新しなければならない。

新しいパリティーデータは、RAID レベル 3 では、2.1 式から、RAID レベル 4、5 では以下の式から算出される。

$$\text{新パリティ} = (\text{旧データ} \oplus \text{新データ}) \oplus \text{旧パリティ} \tag{2.3}$$

この時、パリティの算出に必要な旧データは、その都度、各ディスクから引き出してこなければならない。ゆえに一回の書き込みにつきデータとパリティの各読み書き、すなわち合計 4 アクセスを必要とする事になる [1][2]。RAID レベル 5 の書き込みのタイムチャートを図 2.2に示す。

そのため、パリティ情報を用いる RAID レベル 4、5 は、読み込み速度は冗長情報を保持しない通常のディスクアレイと変わらないものの、書き込みのオーバヘッドが非常に高いと言う問題点を持つ。

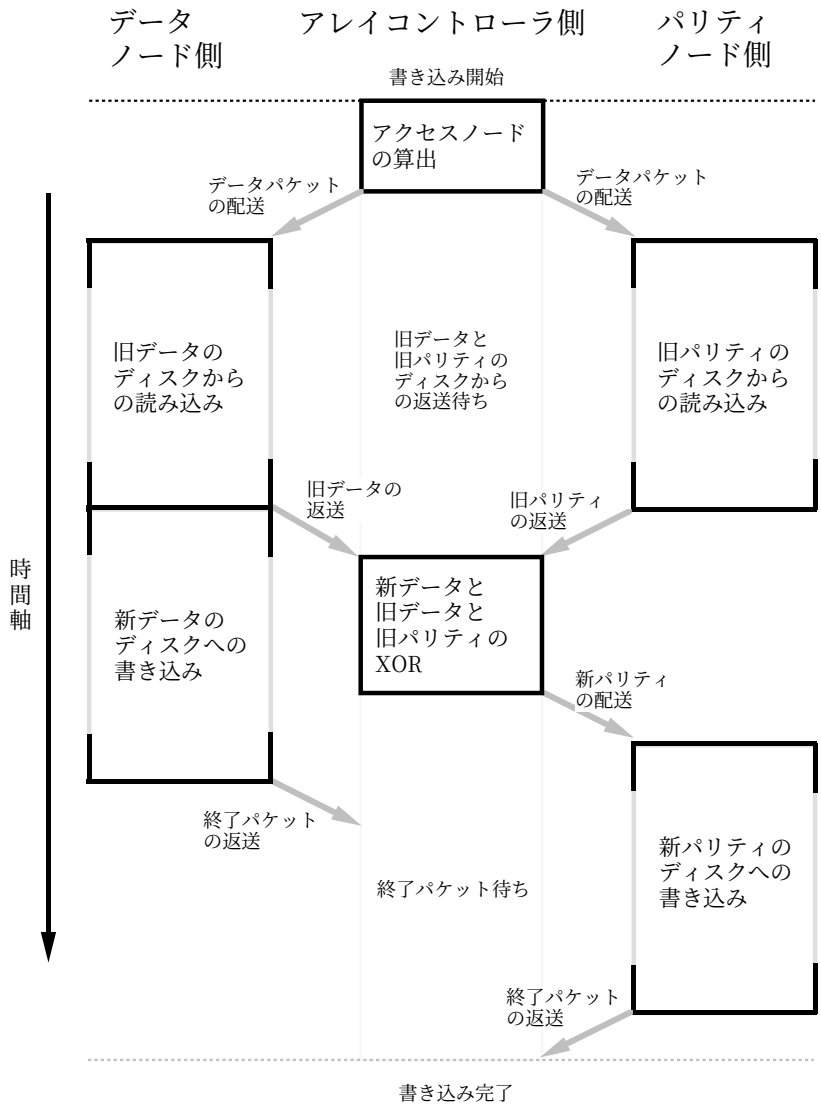

図 2.2: RAID レベル 5 の書き込みのタイムチャート

図 2.3: 2 重キャッシュ付き RAID の構成

このシステムでは同性能のキャッシュを 2 つ搭載し、それぞれのキャッシュに同じデータを置く。各々のキャッシュは、独立した電源供給がなされており、電源障害によって 2 つのキャッシュが同時に故障する確率を低く抑えている。そのため、もし片方のキャッシュ

に障害が発生しても、データはもう 1 つのキャッシュ上に生き残っているので、そのまま処理を続行する事が出来る。しかし、2 重化ディスクキャッシュは、同じデータを 2 台のキャッシュメモリに分けて置かなければならないため、キャッシュのメモリ利用率は悪い。

2 重キャッシュ付き RAID、あるいは単一キャッシュ付き RAID でのアクセスパターンを紹介する。この方式では、キャッシュ上の同一ディスクアドレスデータの更新、すなわちキャッシュ上のデータがヒットしている限り、そのデータのパリティ情報の計算は行なわない。そのため、データがキャッシュにある限りその更新には旧情報のアクセスを必要としないので更新時操作の高速化が期待できる。

実際の、パリティ情報の計算はデータをディスクに移す段階、すなわちデータがミスした場合に行なわれる。この時のアクセスの概念図を図 2.4に、タイムチャートを図 2.5に示す。

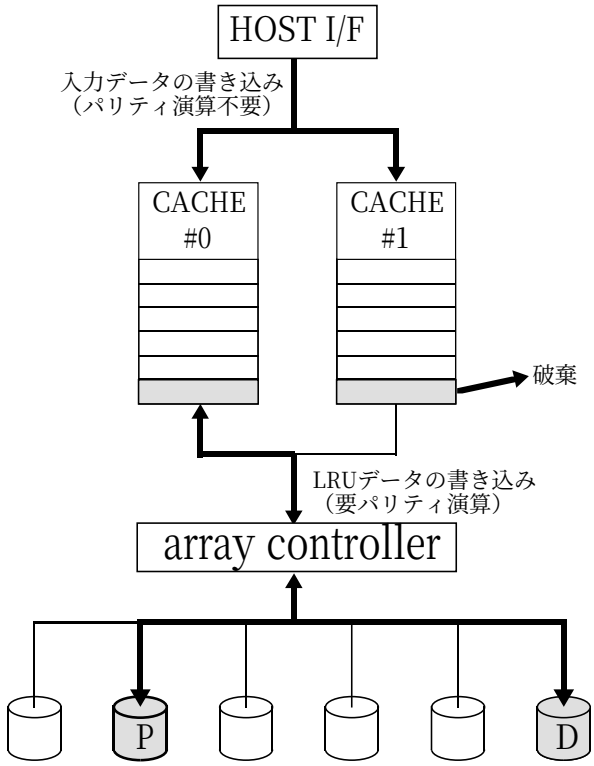

図 2.4: 2 重キャッシュの書き込み・データミス

最初にホストから入力されたデータはキャッシュにそのまま書き込まれる。新しいデータがキャッシュに書き込まれた事で、そこにあったデータはディスクにデステージされな

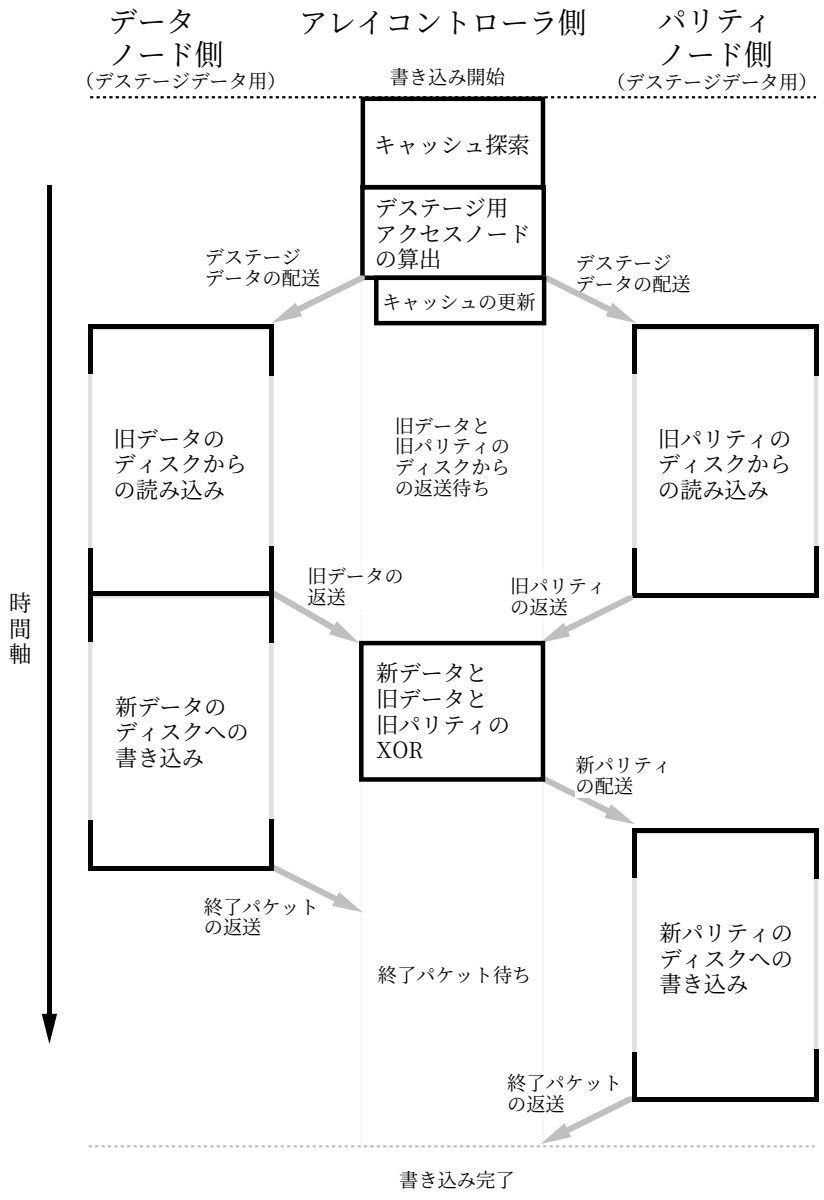

図 2.5: 2 重キャッシュの書き込み・データミスの場合のタイムチャート

ければならない。2 重キャッシュ方式の場合、このキャッシュからディスクに移されるデータに対してパリティ計算を行なう。

この辺りの操作は、まずディスクから古いデータとパリティ情報を読み込んできて、今デステージするデータと排他的論理和を計算して、新パリティ情報を得る。そして、デステージデータと今求まった新パリティ情報をそれぞれディスクに書き込む。

ディスク故障時のディスクへのアクセスは、基本的にキャッシュのない RAID レベル 4、5 と同じである。片方のキャッシュに障害が生じた時は、残った方のキャッシュが代わりに処理を行なう。

第 3 章

分散ディスクキャッシュの仕組みと特徴

2 重キャッシュ付き RAID では、キャッシュの使用とキャッシュデータの 2 重化（ミラー化）によってシステムに高速性と耐故障性を与える事に成功している。本研究では同じく耐故障性を持ったキャッシュを用いる事で、システムの高速化と耐故障性の付与を試みる。ただし、ここではキャッシュに耐故障性を与える方法として、キャッシュメモリを各ディスクノードに分散させ、データのパリティ情報をその時点で持つ事によってキャッシュ上のデータに耐故障性を与える。

3.1 分散ディスクキャッシュ付き RAID の構成

分散ディスクキャッシュシステムは、使用する RAID のディスクドライブと同じ台数のディスクキャッシュを用意し、それを各ディスクコントローラに隣接する形で各ディスクに 1 台づつ割り当てる。各キャッシュは、回線で結ばれており、互いに動作のチェックや隣接するノードのキャッシュデータのアドレス情報等の交換と言ったローカルな通信を行なう事が出来るようになっている。

また各キャッシュは、それぞれが補助電源を持っており、電源障害によるデータの破損を局所的な範囲に抑えるようにしている。分散キャッシュを取り付けた RAID の構成を図 3.1に示す。

次の、ホストからのアクセスがあった際、分散ディスクキャッシュ付き RAID がどのような挙動をするか、追ってみる事で、システムの動作を説明してみたい。

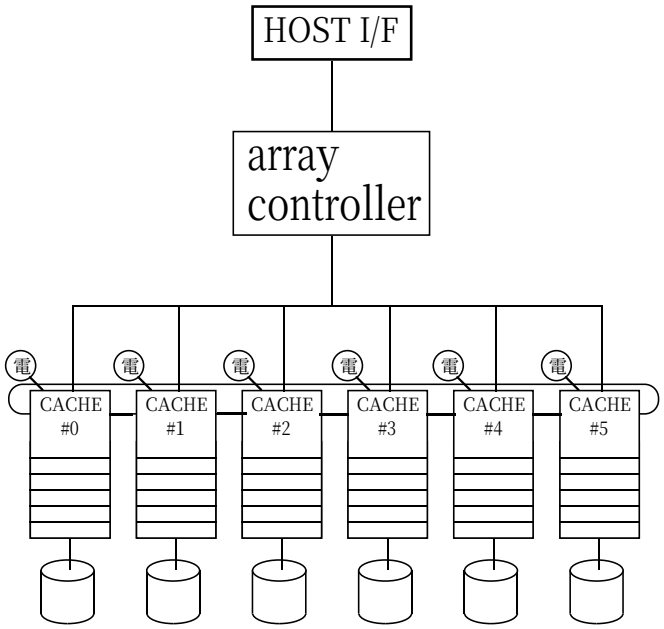

図 3.1: 分散キャッシュ付き RAID の構成

3.2 分散ディスクキャッシュへの書き込み

2 重キャッシュ付き RAID が、キャッシュからディスクにデータをデステージする際パリティ更新作業を行なうのに対して、分散キャッシュ付き RAID ではホストからデータの書き込み命令が送られてきた時点で、そのデータに対するパリティグループのパリティ情報の更新作業を行なわなければならない。更新作業は 2.3 式に従い、旧データの読み込みと旧パリティ情報の読み込み、それとそれらと今ホストから入力された新データとの排他的論理和演算である。

このシステムでは、キャッシュ方式は write-in(write-back) キャッシュを使用しており、キャッシュメモリ上には、ホストから送られてきた最新のデータか、最新のパリティ情報が書き込まれる。キャッシュとディスクドライブは 1 対 1 対応している。あるディスク宛に書き込まれたデータは、対応するキャッシュにまず書き込まれる。

データ更新によってキャッシュに書き込まれたパリティ情報とデータは、いったんキャッシュに入った後は互いに関係なく、それぞれ独自に各キャッシュ内で置換される。そのため、データがキャッシュからディスクにデステージされたがパリティ情報はまだキャッシュに残っている、あるいはその逆といった現象が生じる事になる。

従って、新たに書き込まれたデータに対する、ヒットとミスのパターンは 4 種類ある事になる。

- データとパリティ情報、両方がキャッシュでヒット
- データはミス、パリティ情報はヒット
- データはヒット、パリティ情報はミス
- データとパリティ情報、両方ともミス

以下に、そのパターンの詳細を述べる。

3.2.1 データとパリティ、両方ともにヒットする場合

前に書き込まれたデータに対して、そのグループのパリティ情報とデータがそれぞれキャッシュに残っている場合。

この場合、キャッシュ探索の後、新たに入力されたデータに対する旧データと旧パリティのアクセスは、キャッシュメモリ上から素早く行なわれる。読み込みが終った後、新データは即座にキャッシュメモリ上に書き込まれる。そしてこれらのデータの排他的論理和演算を行なった後、更新されたパリティもまたキャッシュ上に戻される。この時の概念図とタイムチャートを図 3.2と図 3.3に示す。

図 3.3: 分散キャッシュ：データとパリティ、共にヒットした場合のタイムチャート

3.2.2 データはミスしたが、パリティ情報はヒットした場合

データがキャッシュからディスクにデステージされてしまっているが、パリティ情報はまだキャッシュ上に残っている場合。

この場合、古いパリティ情報はキャッシュから素早く読み込んでこれるが、旧データはディスクから読み込んでこなければならない。アレイコントローラは、旧データがディスクから送られてくるのを待っている間に、キャッシュから届いた古いパリティ情報と新データの間の排他的論理和をとっておいて、最終的な演算終了時間の短縮を図る。

旧データが格納されている側のディスクコントローラとキャッシュは、ディスクから旧データを読み込んでアレイコントローラに発送した後、後にキャッシュに書き込まれる新データの置き場所を作るために、キャッシュ内の LRU データをディスクにデステージする。

新データは、ディスクにデステージデータが送られた後、その LRU データのあった番地に上書きされる。この間、アレイコントローラは旧データを受け取って、先ほど行なった古いパリティ情報と新データの排他的論理和の演算結果と旧データとの排他的論理和を行なう。そして新しいパリティ情報が求まったら、キャッシュ上の古いパリティ情報の置いてあったキャッシュ上の番地に新しいパリティ情報を書き込む。

この時の概念図とタイムチャートを図 3.4と図 3.5に示す。に示す。

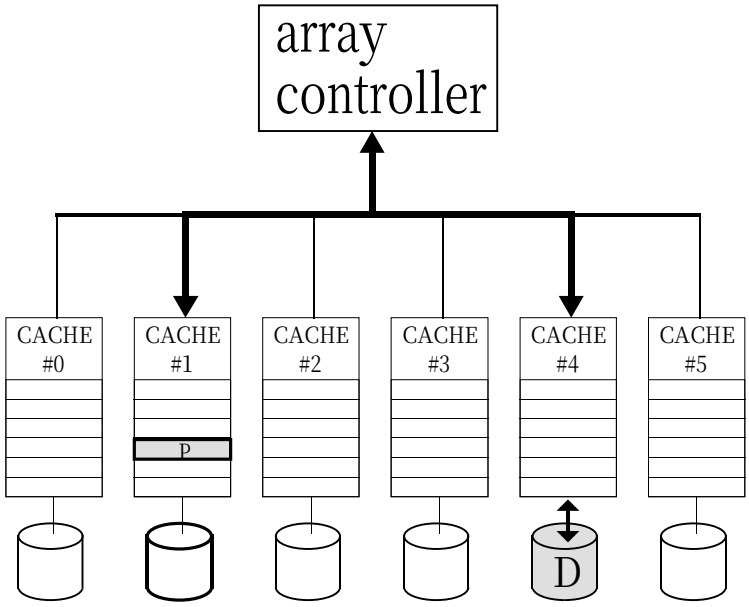

図 3.4: 分散キャッシュ：データはミス、パリティはヒットした場合

3.2.3 データはヒットしたが、パリティ情報がミスした場合

上とは逆に、以前に書き込まれたデータはまだキャッシュ上に残っているが、そのグループのパリティ情報は既にディスクにデステージされてキャッシュメモリ上には残ってない場合。

動作そのものは、データとパリティ情報が入れ替わった以外、基本的に上記のデータはヒットしたがパリティ情報がミスした場合と似たような動作をする。唯一、新しいパリティ情報が演算し終るまでキャッシュへのパリティ情報の書き込みは出来ない点が異なるが、キャッシュへに書き込みは、極めて速いので他に与える影響は小さい。むしろ、新しいパリティ情報の演算が終る前、2 回目の排他的論理演算が始まる前にキャッシュの LRU データの書き込みが行なえる所が優れている。この時の概念図とタイムチャートを図 3.6 と図 3.7に示す。

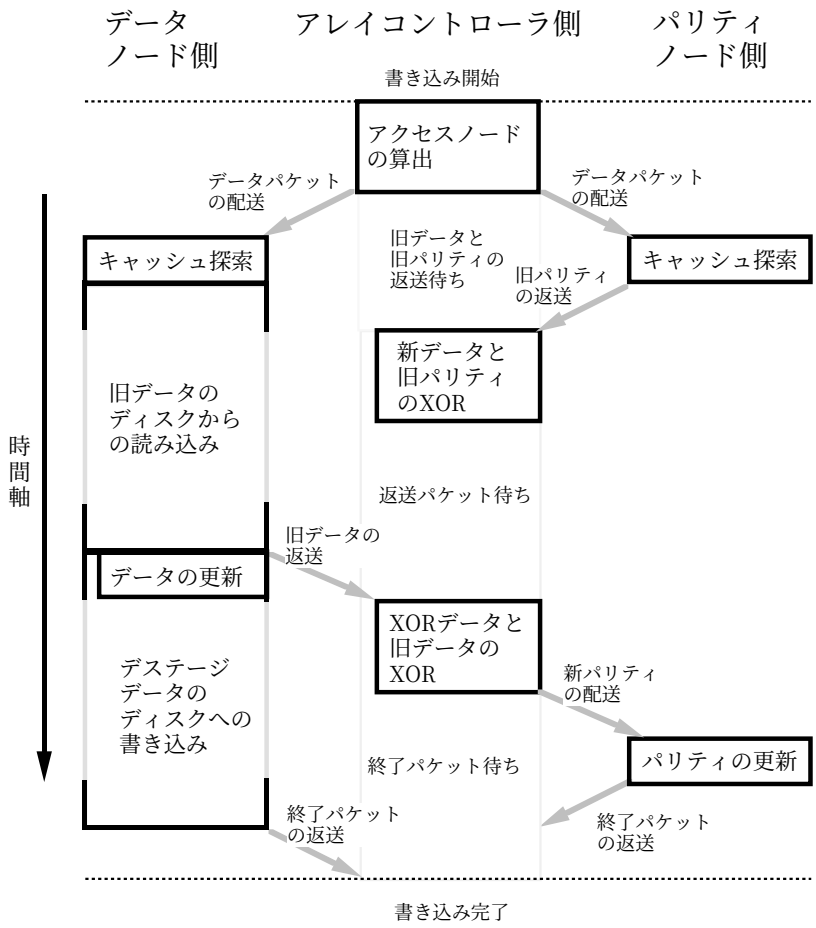

図 3.5: 分散キャッシュ：データはミス、パリティはヒットした場合のタイムチャート

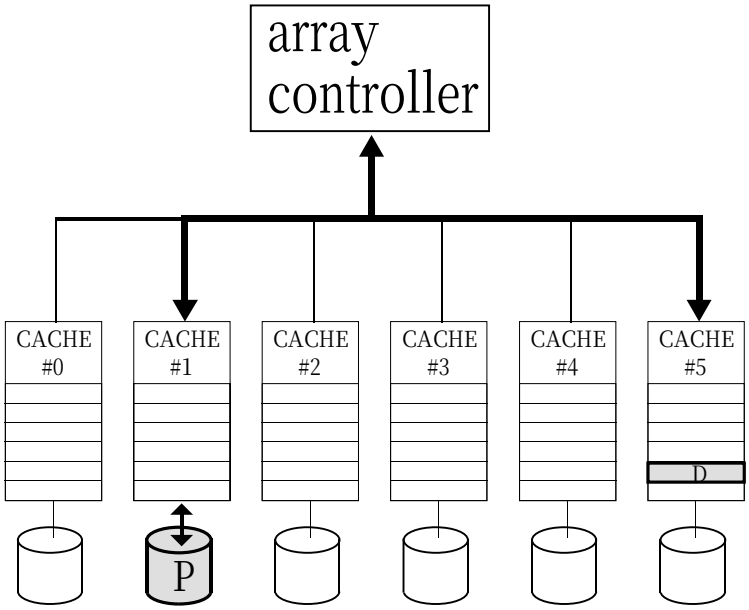

図 3.6: 分散キャッシュ：データがヒット、パリティはミスした場合

3.2.4 データとパリティ、両方ともミスした場合

データとパリティ共にキャッシュに既にない場合、キャッシュ探索の後、両データをそれぞれディスクから読み込んで来なければならない。そして新データとの排他的論理和演算と、新データと新パリティ情報のキャッシュへの書き込みが行なわれる。この時の概念図とタイムチャートを図 3.8と図 3.9に示す。

この際、両キャッシュは、新データあるいは新パリティ情報のキャッシュへの書き込みに備えて、LRU データをディスクにデステージしておかなければならない。ただし、この作業自体は新パリティ情報の算出を待たずに、旧情報の読み込み直後、排他的論理和演算と並行して行なう事が出来る。

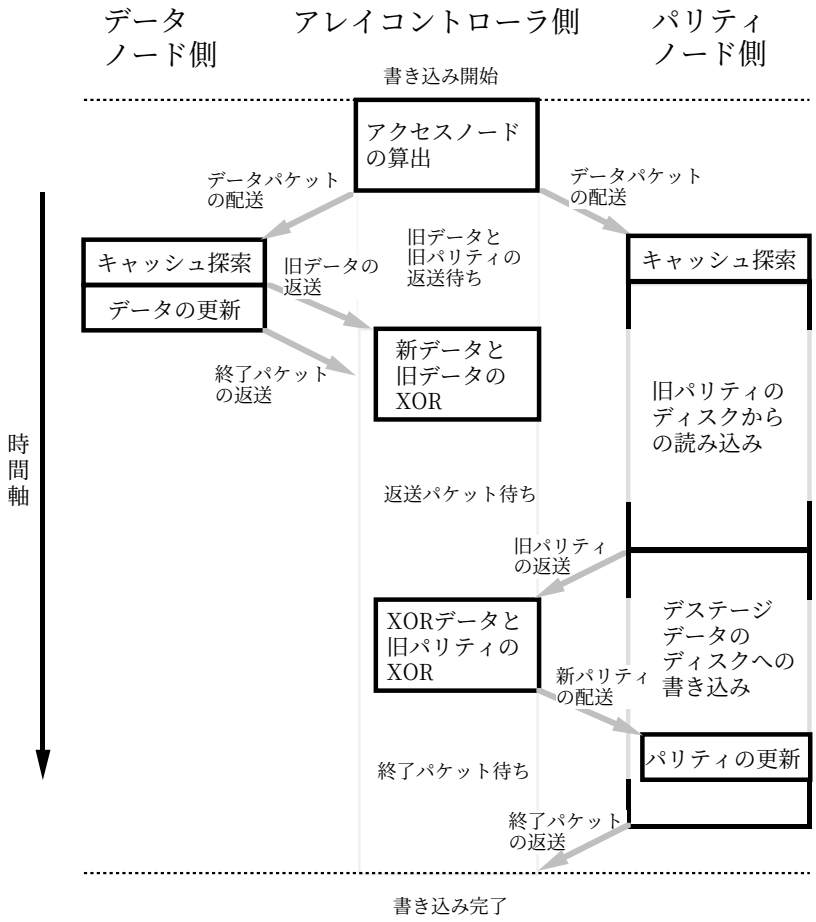

図 3.7: 分散キャッシュ：データがヒット、パリティはミスした場合のタイムチャート

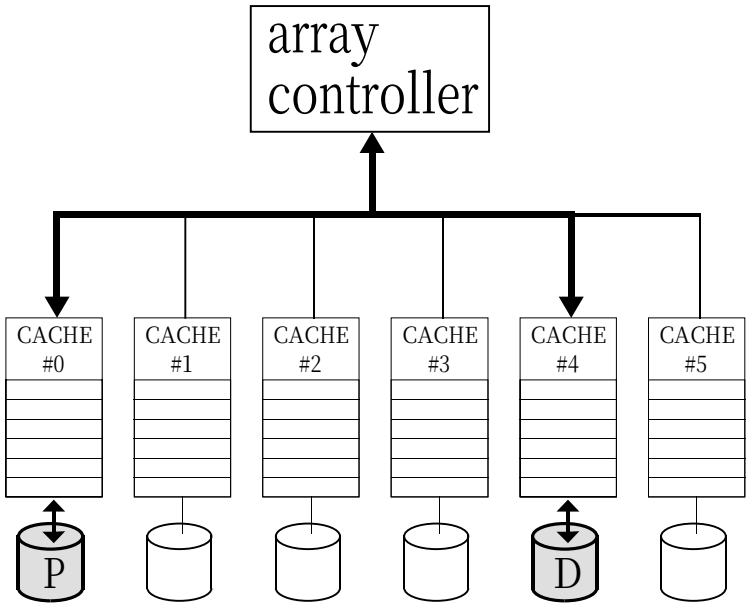

図 3.8: 分散キャッシュ：データとパリティが共にミスした場合

3.3 分散ディスクキャッシュからの読み込み

次に、分散ディスクキャッシュからの読み込み。無故障時の読み込み操作では、パリティ演算を行なう必要がないので、普通のデータの読み込み操作が行なわれる。この際、キャッシュにデータが残っていればそのままキャッシュからデータを読み込んでくるし、ミスすればディスクから読み込んでくる。合計のアクセス時間は、キャッシュ探索時間が加わる分だけ、キャッシュ未搭載の RAID より遅くなる。

なお、データ更新作業に伴って幾つかの処理が互いに並行に実行されているが、処理中の障害によってデータの完全な消失は生じる危険を避けるため、幾つかのデータは各コントローラのテンポラリエリアの保管され、次の書き込みがなされるまでそのまま保持される。

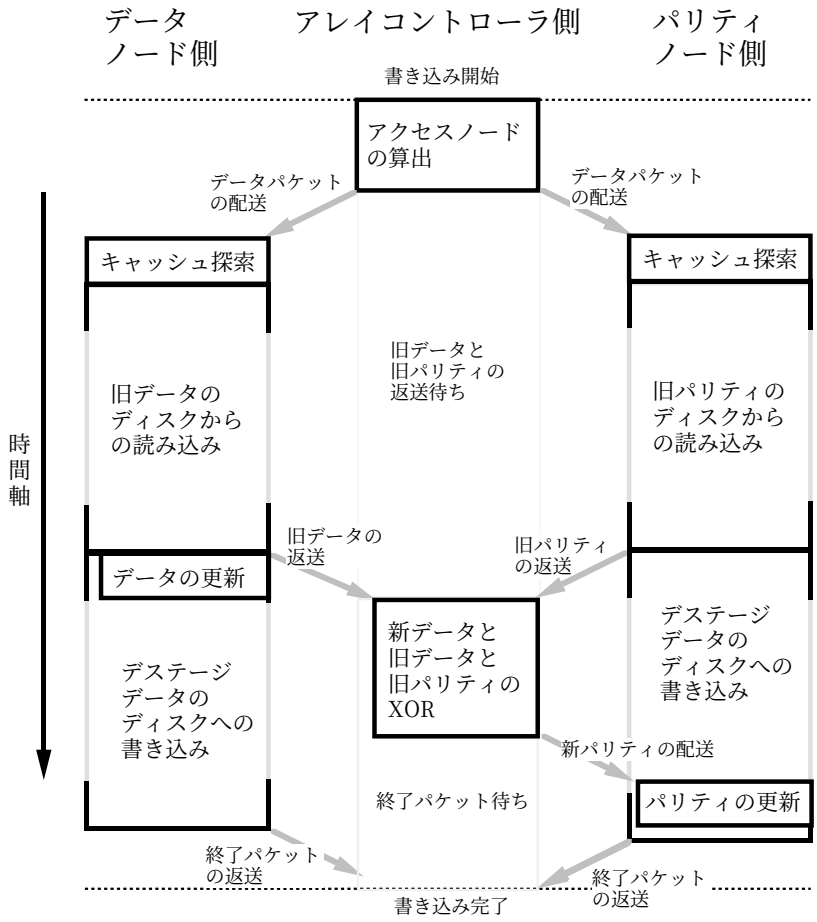

図 3.9: 分散キャッシュ：データとパリティが共にミスした場合のタイムチャート

3.4 故障モデル

分散ディスクキャッシュシステムでは、ディスクドライブに関しては単一ディスク故障まで耐える事が出来る。キャッシュ故障に関しては、1 度の故障で壊れるキャッシュの台数が 1 台までならば、その内容を復元する事が出来る。データ復元後、キャッシュの故障したディスクは、キャッシュなしの普通のディスク装置としてそのまま使用される。

と言う訳で、次にディスクシステムあるいはキャッシュメモリに障害が発生した場合の、データ回復の方法と、故障後の動作を説明する。

3.4.1 ディスク故障時の対応

ディスク故障が生じた時、ディスクコントローラはアクセスの結果がディスクから返ってこない事によって、ディスク故障の発生を認識する。

一般的な RAID レベル 3、4、5 がそうであるように、この場合も壊れたディスクのデータは他のディスクやキャッシュに残っているデータとパリティにいって復元される。復元作業は、読み込みの度に行なわれる。キャッシュが生き残っていれば、キャッシュ上のデータはそのまま読み書き可能である。

書き込みは、ディスク故障が生じてない場合と同じである。（復元されて）読み込まれた古いデータ（あるいはパリティ情報）から、新しいパリティ情報を算出して、そのままキャッシュに書き込む。LRU データはそのまま破棄されるが、パリティ情報あるいはデータそのものが他のディスクやキャッシュに残っているので支障はない。

3.4.2 キャッシュ故障時の対応

キャッシュ故障の認識は、アレイコントローラあるいは各キャッシュとディスクへの入出力を統括するディスクノードコントローラによってなされる。認識の方法は、ディスク故障時と同じく、アクセスに対する結果の返答が戻ってこない事によってなされる。

ディスクとキャッシュへの入出力を統括するコントローラが故障した場合は、そのディスクへのアクセスが不能になるので、以降の操作は、上記のディスク故障時の動作と同じである。ただし、キャッシュも壊れているので、アクセスそのものは全てキャッシュ・ミスした事になる。

キャッシュのみが破壊され、ディスクへのアクセスが可能な場合。キャッシュ上に存在

したまだディスクにデステージされていないデータに対し、ディスク上の該当するデータは少し古く、そのデータをそのまま使用したのでは、その部分のパリティグループの整合性が保てなくなる。そこで消失したデータの復元をまず行なわなければならない。

各キャッシュ上のデータとパリティは、ホストから入力された時点でこそ、それぞれキャッシュの上に存在するが、それ以降のキャッシュ内での置換推移は各自ばらばらである。特定のディスクにアクセスが集中すれば、そのディスクのキャッシュ上のデータは、その相方とは関係なく、さっさとデステージされてしまうであろう。よって、あるキャッシュにデータが存在したからと言って、他のキャッシュにその痕跡があると言う保障はない。しかし、失われた数%のデータを復元するためだけに、アレイグループの全てのパリティの整合性をチェックすると言うのは、手間の掛かる作業である。

そこでキャッシュ情報をよそのキャッシュ上にも残す事を考える。この場合、必要なのはキャッシュ上のデータのアドレス情報だけで残っていれば良い。アドレスさえ残っていれば、他のアレイのデータ照合によってデータの値の復元が可能だからである。復元されたデータは、そのまま故障キャッシュのディスクに直接に書き込まれる。また、壊れたキャッシュに保持されていた他のキャッシュのアドレス情報は、復元作業の後、その壊れたキャッシュのアドレス情報があった場所に再構築される。

そしてデータの復元後は、システムは普通のディスクアレイとして再びアクセス出来るようになる。ただし、キャッシュは壊れているので、該当ディスクへのアクセス速度は、キャッシュ・ミスした場合と等しくなる。

アドレス情報の更新は作業は、データがキャッシュにヒットした場合は、同一アドレスのデータがそのまま上書きされるだけなので不要で、実際はデータのデステージの都度ごとに行なわれる。更新作業は、ディスクへの読み込みが行なわれている間になされるので、通常のアクセス時間に影響を及ぼす事はない。

アドレス情報の保持の方法は二通り考えられる。

- ディスクのビットマップを隣接するキャッシュ上に構築する方式。該当アドレスのデータがキャッシュにあるかないかでビットデータを変える。更新の際に必要な情報はディスク上のアドレスのみでよい。

- キャッシュに入っているデータのディスクアドレスとキャッシュ上での位置を保持する方式。キャッシュのマップ情報を隣接するキャッシュに構築する。更新に必要な情

報は、ディスク上のアドレスとデータが格納されるキャッシュ上のアドレスの 2 つである。

キャッシュ障害が生じた際は、隣接するキャッシュに保持されたこのマップを参照しながら、失われたデータを復元する。キャッシュのアドレス情報を隣接キャッシュに構築する方法の概念図を図 3.10に示す。

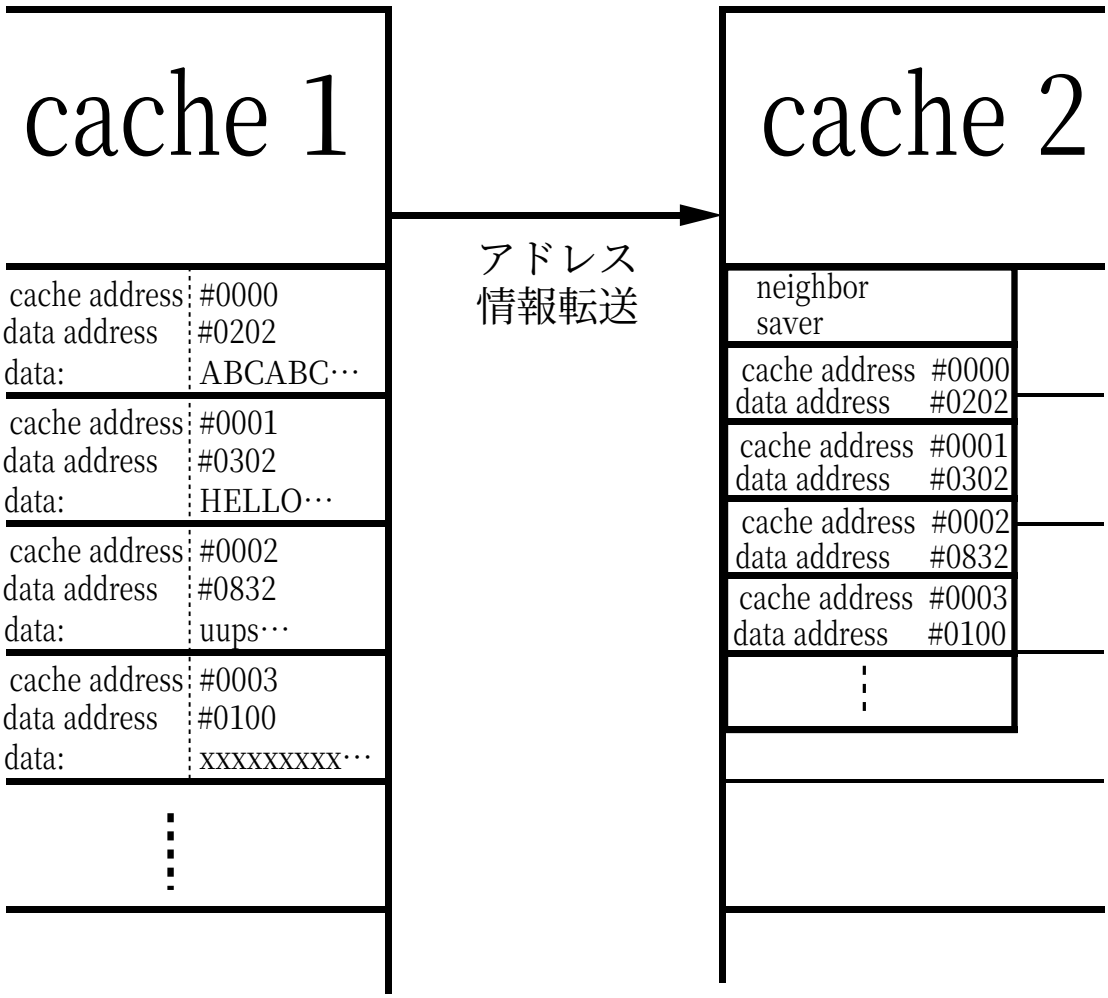

図 3.10: キャッシュのアドレス情報を隣接キャッシュに構築する

第 4 章

実験システム

分散ディスクキャッシュに実験システムとして、実際のアクセス時間等を測定できるシステムを構築し、各種データの測定を行なう事にする。

この章ではトランスピュータを使用した実際にディスクアクセス可能な分散ディスクキャッシュ実験システムの構成を説明する。まず、ハードウェア構成の概要について述べ、その後実際のインプリメンテーションについて説明する。

4.1 ハードウェア構成

実験システムは、本研究室で開発された DR-net で用いられたディスクアレイ装置上に構築した。ディスクアレイ装置は筐体内に五つの VME サブラックを有し、各 VME サブラックには 5 台づつのプラグインユニットが実装されている。各プラグインユニットは以下の装置で構成されている。[5][6]

- inmos T805 (25MHz) を使用した CORAL HPT04 SCSI-2 TRAM (SCSI コントローラ)
- C004 リンクスイッチチップ搭載 IMS B014 VMEbus ボード
- Quantum Go·Drive 120S (2.5" 120MB SCSI ハードディスクドライブ)
- 電源装置
- ケース

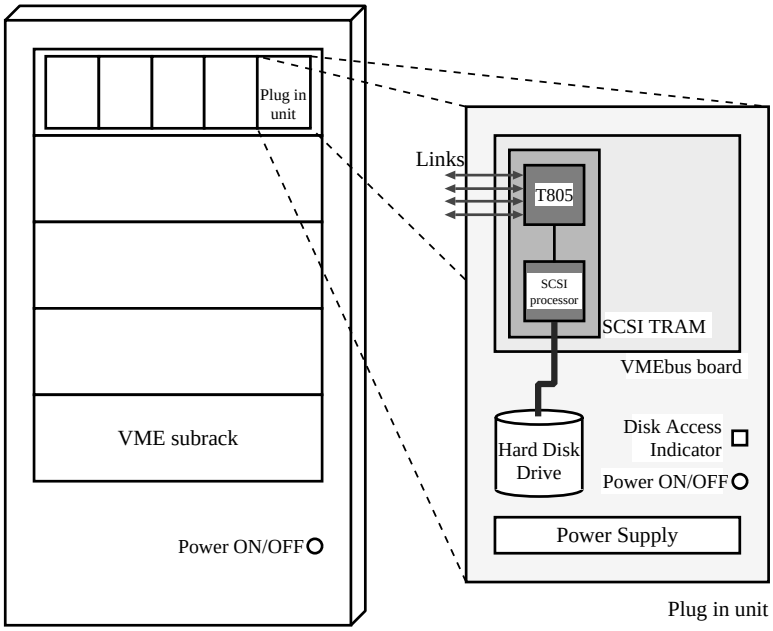

図 4.1: ディスクアレイ装置のハードウェア構成

収納されている Quantum 製のハードディスクドライブは平均シークタイム 17ms、平均回転待ち時間 8.3ms、最大転送速度 4MB/s をサポートしている。

occam でディスクの読み書きを行なう際は、ブロック単位でアクセスを行なう。1 ブロックは 512 バイトからなり、このハードディスクドライブは 24 万ブロックの区域から構成される。本研究、このうち 1 台あたり 4 万ブロック＝ 20MB の区域を実験用に使用した。

各々の T805 はリンクを介して隣接ノードと接続されて、5×5 ネットワーク構造を構成している。そして外部インタフェース用のルートノードとして、さらにもうひとつ T805 が接続されており、ルートノードはホストコミュニケーションのために使用される。実験システムは IMS B300 TCPlink を介してイーサネット経由で利用することができる。図 4.2にディスクアレイ装置のネットワーク構成を示す。

ノード間は公称値 20Mbits/s で相互通信が可能となっており、VMEbus ボード上の IMS C004 リンクスイッチチップを使用した実測値は片方向約 1.1MBytes/s(8.8Mbits/s) となっている。

本研究では、このうち図中で網掛けされたノードを使用した。各ノード上に動作する処理と、その接続関係を以下に示す。右肩の数値はノード番号を意味する。ノード番号 44

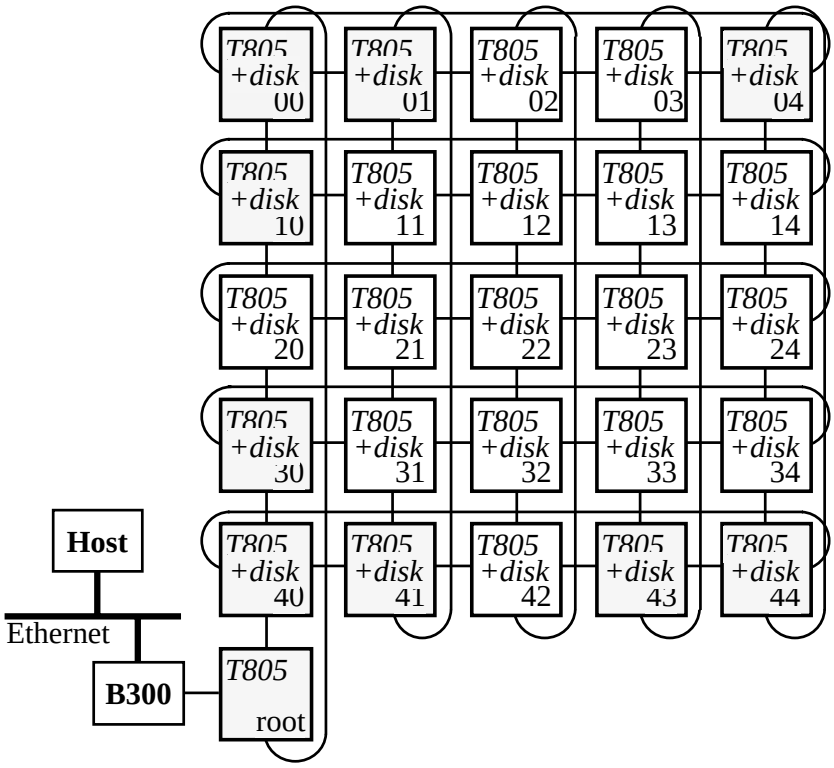

図 4.2: ディスクアレイ装置のネットワーク構成

図 4.3: 実験に使用したノードの構成

4.2 ソフトウェア構成

実験システムでは、ファイルをパケット単位に分割して各ノードへ配送するようにした。この章では、パケットの構成法、各ノード上で働くアレイ制御プロセス、中継プロセス、キャッシュ制御プロセスなどについて述べる。なお、すべてのプログラムは occam で記述した。

4.2.1 パケットプロトコル

すべてのノードはネットワーク内の位置に応じた自身のアドレスを持っている。パケットはそのアドレスを元に、発信ノードから目的ノードへと配送される。ノード間を配送されるパケットは

目的地アドレス	オペレーション	ディスクページ	データ

のようなフォーマットを持ち、目的地アドレスの情報を元に中継-ノードで分岐しながら、目的地へと配送されて行く。

発信地に届いて、オペレーションに従って処理がなされた後、読み書きのデータとディスクアクセスの結果などが、ホスト-ノードに返送される。この際に使用される返送パケットは、以下のようなフォーマットを持ち、中継-ノードを辿りながらホスト-ノードへと返送されてゆく。

発信地アドレス	アクセスデータ	アクセス結果	計測時間

中継プロセスは、これらのパケットをホスト側から目的地側へ、あるいは目的地側からホスト側へ受渡しを行なうだけで、内容の変更は特に行なわない。

4.2.2 ホストノードのプロセスとチャネル

ホスト-ノードには、ホストプロセスとアレイ制御プロセスとが設置されている。ホストプロセスは、画面表示や実験用の各種乱数データの作成、キャッシュノードから返ってきた計測データのを集計などを行なう。アレイ制御プロセスは、ホストから呼び出されるプロセスで、ホストから与えられたデータの入出力命令を元に、実際の RAID 操作を行なう。

図 4.4にホストノードの構成を示す。

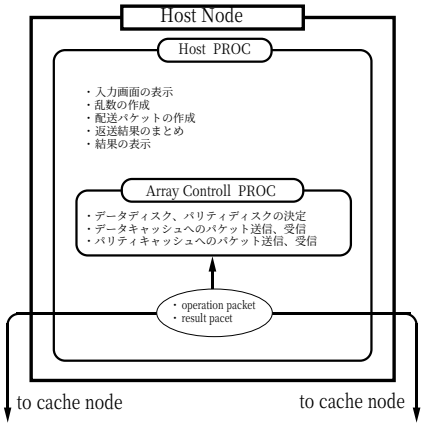

図 4.4: ホストノードの構成

アレイ制御装置は、ホストが指定してきた論理ディスク番号とアドレス番号から、実際にアクセスする物理データディスクドライブとパリティディスクドライブの位置を算出す

る。RAID レベル 5 型での物理データディスクと物理パリティディスクの位置は、以下の式で導出される。使用するディスクドライブの台数を N とすると、

パリティディスク位置 = アドレス % N
データディスク位置　 =（パリティディスク位置 + 論理ディスク番号 + 1）% N

ここで指定可能な論理ディスクは 0 番から N-2 番までで、物理ディスクドライブは 0 番から N-1 番まで割り振られる。RAID レベル 4 型では、パリティディスクは常に決まっており、N-1 番ディスクをパリティディスクに固定すると、0 番から N-2 番までの論理番号の指定がそのまま物理ディスクのドライブ番号になる。

4.2.3　パスノードのプロセス

装置構成上、1 台のホストノードと 6 台のキャッシュノードを直接に結線する事は出来ないので、パケットの仲介用ノードとしてパスノードを設定した。パスノード自体は、データの加工は特に行なわない。パスノードは 2 種存在し、1 つはホスト側とキャッシュ側の入出力信号線が 1 対 1 対応しているもので、もう 1 つはホスト側 1 に対し、キャッシュ側 3 方向のものである。後者は、配信パケットの目的地アドレスを参照して、3 つの方向のどれかにパケットを発送する。キャッシュ側から返送パケットを受信した時は、単にホスト側信号線にパケットを送る。

図 4.5にパスノードの構成を示す。

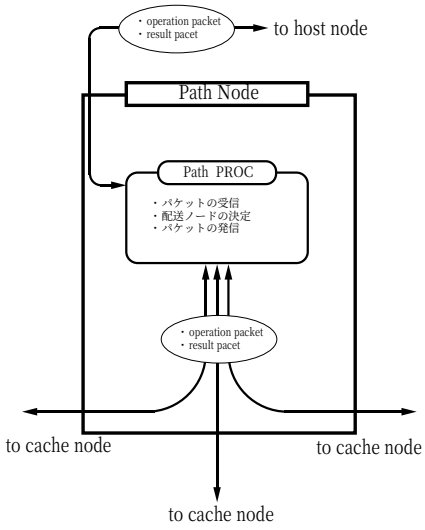

図 4.5: パスノードの構成

4.2.4 キャッシュノードのプロセス

キャッシュノードは、基本的なディスクアクセスを行なうディスクプロセスと、ディスクキャッシュ処理を行なうキャッシュ制御プロセスからなる。キャッシュノードは、キャシュ内のデータ探索の他、ホストが排他的演算を行なっている間にデステージデータのディスクへの書き込み命令をディスクプロセスに出す事が出来る。また、実験開始の際は、ディスクから適当なデータを読み込んできて、キャッシュに格納する。そして実験終了の際には、キャッシュデータのディスクへの書き込みを行なう。

キャッシュ方式はとしては、リンクリストによるフルアソシアティブ方式で LRU 置換を行なう方法と、入力ブロック数を元にしたハッシュ関数によるキャッシュ番地上へのダイレクトマッピング方式のどちらかを任意に使用できるようにしている。本研究で用いたキャッシュは、occam で記述したソフトウェアによるキャッシュである。そのため、フルアソシアティブ方式は速度が遅く、キャッシュ容量を増やすと、キャッシュ探索時間がかかるようになる。ヒット率の測定用には問題はない。ダイレクトマップ方式では、ヒット率は低いが、キャッシュ探索の速度は速く、容量を変えても掛かる時間は少ない。[7][8]

図 4.6にキャッシュノードの構成を示す。

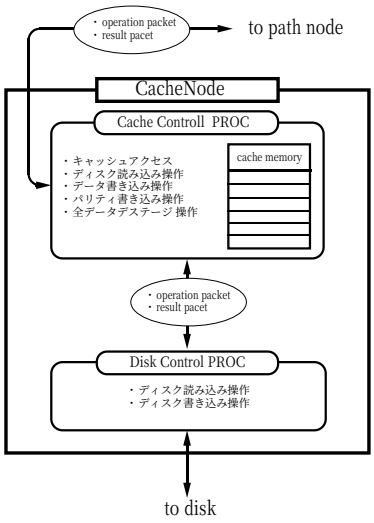

図 4.6: キャッシュノードの構成

4.2.5 ノードの動作とパケットの受渡し

ホストからの命令が、データの読み込みだった場合。まず、アレイ制御装置は入力された論理ディスク番号から、物理ディスク番号を決定し、そのノードにオペレーションとして data-read を指定した配送パケットを発信する。そして物理データディスクからのデータの入った返信パケットが送られてくるのを待つ。

ホストからの命令が書き込みだった場合。アレイ制御装置は、物理ディスクとパリティディスクを決めた後、データ側ノードに対して data-write のオペレーションを、パリティ側ノードには parity-write のオペレーションを指定した配送パケットを送る。

両データを送った後、アレイ制御装置はどちらかのノードからの旧情報を含む返信パケットが届くのを待つ。旧情報が送られてきたら、パリティ、データどちらに関わらずホストから入力された新データとの排他的論理和演算を行なう。そしてもう片方のデータが送られてきたら、その演算結果との間の排他的論理和を行なう。そしてその演算結果をパリティ側のノードに対して再び発信する。

パリティノードは旧パリティ情報をホストノードに返信した後は、アレイ制御装置からの新パリティ情報の受信待ちつつ、キャッシュの LRU データのディスクへの書き込みを行なう。そしてアレイ制御装置から新パリティ情報が届いたら、それをキャッシュに書き込む。そしてディスクへの書き込みが成功したならば、アクセス結果にキャッシュでの

ヒットの有無を記載して、ホストに2度目の返信パケットを送る。もし、ディスクへの書き込みでエラーが生じた場合は代わりにアクセス結果に発生したエラー番号を記載してホストに返信する。

ディスクノードの場合は、パリティノードと異なり新パリティを待つ必要がないので、旧データを返信後、そのまま最初にホストから送られてきたデータのキャッシュへの書き込みと、LRU データのディスクへの書き込みを実行する。そしてパリティ側と同様に、ディスクアクセスの結果にヒットの有無かディスクでのエラー番号を記載して返送パケットをホストに送る。

ホストは、パリティ側からとデータ側からの返送パケットを受信し、そのアクセス結果がエラーでなければ、ホストプロセスに書き込み完了と伝える。

そして最後に、ホストから実験終了の命令が届いたら、アレイ制御装置は各ディスクノードに end のオペレーションの配送パケットを送る。ディスクノードは、キャッシュのデータを全てディスクに書き込んだ後、アクセス結果に完了のマークを付けて、アレイ制御装置に返す。書き込みに失敗した場合は、エラーマークを付けて返す。アレイ制御装置は、全てのディスクノードから完了したとのパケットを受け取った後、ホストプロセスに全完了を伝える。

4.3 使用モデル

分散キャッシュモデルを適用する RAID モデルとしては、RAID レベル 4 と 5 をそれぞれ製作した。この実験システムでは、各ノードのキャッシュ容量をトランスピュータの記憶容量の範囲内で自由に設定する事が出来る。

また、比較モデルとして、キャッシュ無し RAID レベル 5 と、2 重キャッシュ付き RAID レベル 5 を作成した。

2 重キャッシュ部分はホストノード上にプロセスとして構築された。ホストノードから出された読み書きの命令は、一度キャッシュプロセスに渡たされる。データがヒットした場合は、そのままホストに結果が返される。ミスした場合は、命令が読み込みならば、命令はそのままアレイ制御プロセスに送られる。書き込みだった場合は、キャッシュの LRU データのディスクへの書き込み命令がアレイ制御プロセスに渡された後、パリティ計算などが行なわれる。

なお今回の実験は無故障時のデータ集計を目的としたので、比較モデルとしての 2 重

キャッシュは、片側のみキャッシュメモリを実装している。

4.4 計測データと使用データ

この実験装置で計測可能なデータを、以下に示す。これらは、ホストノードか各キャッシュノードで計測される。

- キャッシュ探索時間
- ディスクからの読み込み時間
- ディスクへの書き込み時間
- RAID システムとしての読み込み時間
- RAID システムとしての書き込み時間
- データとパリティ、それぞれのヒットの可否

他にホストからの書き込み操作の際、パリティ側になったノードはホストに旧パリティを返送してから新パリティが返ってくるまでの時間を計測してホストに返している。以上のデータはミリ秒単位でそれぞれ計測集計される。

ホストが送り出す指令としては、読み込みか書き込みかの命令、論理ディスク装置番号、ディスクアドレス（ブロック）、データがある。これら出力データの作成には occam の乱数命令を用いた。それとディスクアドレスには、普通の乱数方式とは別に出力に疑似的に 3 乗の偏りを持たせた乱数を用いた。

偏り乱数を用いたディスクブロックの指定方法について。この場合、0 から 39999 まであるディスクブロックを 200 ブロック置きに 200 の区間に分ける、次に、出現分布に偏りがある乱数を 0 から 199 までの値から選び出し、区画番号とする。そして区画内ではランダムに値は取り出される。偏り乱数は、基数に対して、3 乗倍の出現率を持つ。例えば、区画 100 番が選ばれる可能性は、区画１番より 100^3 倍大きい。

第 5 章

評価と検討

この章では、4 章で紹介した分散キャッシュ付き RAID 実験システムに基づいた性能測定結果を報告する。そしてキャッシュ未搭載な通常の RAID の測定用モデル、2 重ディスクキャッシュ付き RAID の測定用モデルとの比較を行なう。また、キャッシュ方式としては、ソフトウェア式で動作するフルアソシアティブ方式とダイレクトマップ方式をそれぞれ用い比較を行なった。前章でも触れたが、フルアソシアティブ方式のキャッシュは高いヒット率を期待でき、ダイレクトマップ方式のキャッシュは高速な応答時間が期待できる。

以上を基に。RAID で問題となる書き込み性能についての実験を中心に、分散キャッシュ付き RAID 方式の特性と優位性を確認してみる。

なお、本実験では特に断らない限り、以下の値を用いて測定を行なった。

使用パラメータ

使用ディスク数	6 台
ディスク 1 台辺りの使用ページ範囲	0〜3999 ブロック
1 ブロックのサイズ	512 バイト
ディスクアクセス	ランダム
ブロックアクセス	出現分布に偏りのあるランダム
サンプリング回数	読み書き合計 5 万回
読み書きの出現比率	50:50（それぞれランダムに混在して出現）

図 5.1: 分散キャッシュRAID5：キャッシュ容量を変えた時の各ヒット率の変化

RAID レベル 5 方式は、各キャッシュにデータとパリティ情報が混在して格納されるのに対し、RAID レベル 4 方式では両方がそれぞれ別々のキャッシュに格納される。フルアソシアティブ方式の分散キャッシュ付き RAID レベル 4 での、システム全体の総キャッシュ

図 5.2: 分散キャッシュRAID4：パリティキャッシュの容量を変えた時のヒット率の変化

この場合、データとパリティ情報の同時ヒット率は、パリティキャッシュの容量が増えるに従って増加し、システムに対して、パリティキャッシュの容量が 45 %から 50 %の付近で最高値に達した後、データのヒット率とほぼ同じ割合で減少していった。最高値の値は、フルアソシアティブ方式の分散キャッシュ付き RAID レベル 5 のデータとパリティ情報の同時ヒット率とほぼ等しかった。

この事から、RAID レベル 5 型の分散キャッシュ内での、パリティ情報とデータの住み分けも、ほぼ 40 から 50 %付近になる事が推測できる。6 台のディスクからなるこの実験システムでは、パリティ情報はデータに対して 5 倍高い参照確率を持つ。よって、データとパリティ情報の同時ヒット率が、データのヒット率とほぼ等しく、パリティ情報のヒット率の約 5 分の 1 である図 5.1は、同時ヒットと言う点では、かなり良いヒットの仕方をしていると言える。

次に、フルアソシアティブ方式の替わりにダイレクトマップ方式を用いた分散キャッシュ付き RAID レベル 5 の、同様にキャッシュ容量を増やしていった場合の各ヒット率の変化

図 5.3: 分散キャッシュRAID5：キャッシュの容量を変えた時のヒット率の変化。ダイレクトマップ方式

この場合、データとパリティ情報の同時ヒット率は、データのヒット率の約 45 %、パリティ情報のヒット率の約 18 %、フルアソシアティブ方式の同時ヒット率（図 5.1）と比べた場合は約半分であった。分散キャッシュでは、データとパリティが別々のキャッシュに保持されるため、両者のデステージのタイミングが大きくずれれば、それだけヒット率も下がる事になる。従って、デステージメカニズムの選定は大切である。この点において、ダイレクトマップ方式はフルアソシアティブ方式に対して劣っているといえる。

図 5.4 は、2 重キャッシュと分散キャッシュ付きの RAID レベル 5 の、フルアソシアティブ方式とダイレクトマップ方式のヒット率（分散キャッシュは同時ヒット率）を比較したものである。

この図から、分散キャッシュRAID レベル 4,5、そして 2 重キャッシュ、これらにフルアソシアティブ方式を用いた場合、ホストからの書き込み命令に対して、3 つの方式の間にヒット率の差が小さい事が判る。これは分散キャッシュもデータと同じ大きさのパリティ情報をキャッシュに書き込まなければならないため、実際にキャッシュ内でデータの占める割合は約 50%、同じデータを 2 つ保持する 2 重キャッシュ方式と同程度になるからで

図 5.4: 分散キャッシュと 2 重キャッシュの書き込み時のヒット率

ある。

5.1.2 アクセス速度

フルアソシアティブ方式を用いた場合、RAID レベル 4、5 では、分散ディスクキャッシュと 2 重化ディスクキャッシュの間に、ヒット率の差が生じない事が判明した。

3 章で述べたように分散ディスクキャッシュはキャッシュを各ディスクノードに分散して設置するため、処理の分散化による 1 回の作業時間の短縮が期待できる。

と言う事で、今度は各モデルの総キャッシュ容量を 2000 ブロック=1MB に固定した状態でのデータの書き込みに対する各処理ごとの応答時間を見ていく事にする。2 章と 3 章で触れたように、分散キャッシュにはヒットの仕方に 4 種類、2 重キャッシュの場合はヒットとミスの 2 種類のアクセスパターンが存在する。そのそれぞれについて、１つづつ見ていく事にする。

この論文で説明の順番については、アクセス速度に関する分散キャッシュの測定結果を見る前に、まず最初に比較に用いたキャッシュ未搭載モデルのデータと、2 重キャッシュモデルのデータの紹介から行なう事にする。

キャッシュなし RAID5 の場合

このモデルへのホストからの書き込みがあった際の、測定箇所を図 5.5に、測定結果を表 5.1に示す。

このモデルはキャッシュを持たないので、他のキャッシュ使用モデルよりも数 ms 程速く作業を開始できる、

このモデルではディスクアクセスに 21ms と、他の方式に比べて、若干低目の値ではあったが、各ディスクノードがディスクにアクセスする時間に差はないので、そこで生じる時間の差はディスク性能上の誤差であると考える事にする。

2 番目の計測値は、パリティノードが読み込んだ旧パリティ情報をホストに送り出してから、ホストが論理演算処理を施して、再びパリティノードに新パリティ情報が返ってくるまでの応答時間である。なお、ディスクのアクセスはパリティノード、データノード別々に行なわれて、論理演算はとりあえず先に届いた方から行なわれるので、この値がそのまま論理演算時間に直結するとは限らない。しかし、分散ディスクキャッシュの場合は、この計測時間 2 に費やす時間を短縮する事で 1 回の処理時間を短縮させる事に成功しているので、重要な値であるといえる。このモデルでの計測値は 8ms で、2 回の論理演算の時間としては概ね標準的な値であると言える。

3 番目の計測データは、ディスクへの新情報の書き込み時間である。この実験装置で用いたディスクドライブは、バッファを搭載していて書き込み速度の向上を図っている。このバッファは読み込み時には使用されない。書き込み速度は読み込み速度より、4、5 倍近く高速化している。ただし、バッファ容量が小さいらしく、ディスクノードレベルで短期間で 5 回程度の連続した書き込みがなされると、読み込み速度なみに遅くなる事が確認されている。今回の実験ではホストから 1 回 1 ブロックの時間的に長めの間隔でアクセスを行なっているため、そこまで遅くなる事はない。と言うわけで、このモデルでの計測された書き込みに 4〜5ms と言う値は、他のモデルと比べても納得出来る値である。

このモデルでのシステム全体の総アクセス時間、すなわちホストが書き込み命令をアレイコントローラに出してから終了の合図をホストが受け取るまでの時間は、38ms かかっている。計測時間 1、2、3 の合計が 34ms。この場合、差の 4ms はパケットのノード間での転送処理その他の処理に掛かった時間と考える事にする。

表 5.1: キャッシュなし RAID5 の場合

図 5.6: 2 重キャッシュ：ヒットした場合の測定箇所

表 5.2: 2 重キャッシュ：ヒットした場合

2 重キャッシュ：ミスした場合

今度は 2 重キャッシュがミスした場合の説明を行なう。このモデルでの測定箇所を図 5.7 に、測定結果を表 5.3に示す。

まず、キャッシュ探索に、フルアソシアティブ方式で 6ms、ダイレクトマップ方式で 1ms 未満かかっている。フルアソシアティブ方式は、キャッシュメモリをすべて探索しているので非常に遅い。

キャッシュ探索後の処理は、キャッシュ未搭載の RAID モデルとほぼ同じ。ただし、この際ディスクに書き込まれるデータは、ホストから入力されたデータではなく、キャッシュ内の LRU なデステージデータである。キャッシュ更新はほとんど瞬時に、しかも、旧情報の読み込みと並行して行なえるので、トータルなアクセス時間に影響を及ぼす事はない。

計測結果は、その予想を裏付けるものが得られた。

従って、2 重化キャッシュのミス時の総アクセス時間は、キャッシュ探索時間＋キャッシュ未搭載 RAID と考えて十分である。

表 5.3: 2 重キャッシュ：ミスした場合

分散キャッシュ：両方ヒットした場合

今度は分散キャッシュの説明を行なう。このモデルは 4 種類のヒットかミスかのパターンがある。最初は、ホストからのアクセスに対して、データ、パリティどちらもキャッシュにあった場合、つまり同時ヒットした場合をみる。表 5.4に各動作に費やした時間を示す。図 5.8に測定した処理の時間的位置を示す。

まず、アレイコントローラはホストからの命令を基にデータノードとキャッシュノードの位置を算出し、各ノードの “旧情報送レ” のパケットを発信する。ちなみにこの 2 重キャッシュでのヒットには、この操作にあたる処理は存在しない。

各ノードは受け取ったパケットを基に各自のキャッシュの探索を開始する。キャッシュ探索に費やした平均時間は、フルアソシアティブ方式で 3ms、ダイレクトマッピング方式で 1ms 未満。フルアソシアティブ方式に探索時間が、2 重キャッシュの場合の 3 分の 1 になっている。システム全体の総キャッシュ容量は 1MB=2000 ブロック。よって、6 台ある分散キャッシュ方式の場合、1 台辺りのキャッシュ容量は 333 ブロック。容量自体、2 重キャッシュの 3 分の 1 なので、3 分の 1 の時間で済む事は妥当な値と言える。

データ、キャッシュ共にヒットしたので、ディスクアクセスはなし。ヒットしたデータはそのまま間を置かずにホストに送り返される。

旧情報が、ホストノードのアレイコントローラに届いたら、論理演算が開始される。論理演算は、旧データ、旧パリティ情報のどちらかが戻ってきた時点で開始されるが、データが戻ってくるまでの時間が短いので、この場合、その時間的差はほとんど生じないものと考えられる。従って、計測時間 3 の” ホストに旧パリティ情報を送った後、新パリティ情報が戻ってくるまでの時間”…この実験での 9ms…は、2 回の論理演算時間と 2 回のノード間のデータ転送時間の和であると推定される。

この場合の総アクセス時間は 12〜13ms 掛かっている。この値はキャッシュ未搭載の RAID のアクセス時間の 3 分の 1 近い値である。しかし、2 重キャッシュ方式がヒットした時のアクセス時間よりは大きい。従って 2 重キャッシュ方式と比較した場合は、分散キャッシュ方式のヒットした事によるアクセス時間短縮の利得は小さい。

表 5.4: 分散キャッシュ：データとパリティが両方ヒットした場合

分散キャッシュ：データがヒット、パリティがミスした場合

分散キャッシュの 2 番手は、データがヒットしたが、パリティ情報はミスした場合である。表 5.5に各動作に費やした時間を示す。図 5.9に測定した処理の時間的位置を示す。

各ノードにホストからのパケットが届けられた後にキャッシュ探索に費やした平均時間は、フルアソシアティブ方式で、ヒットしたデータ側で 1ms、ミスしたパリティ側で 2ms。ダイレクトマップ方式では、どちらも 1ms 未満。

データノードはヒットした旧データをすぐに返す。ホストノードはそれを受信して、論理演算を早速開始する。その間、パリティノードはディスクへの旧データの読み込みを行なっている。読み込みには 21〜22ms 掛かっているから、その間に旧データと新データとの間の論理演算は終了してしまう。

読み込みが終ったら、パリティノードはホストに旧パリティ情報を送り返し、新パリティを格納するため、LRU なデータをディスクにデステージする。この間、アレイコントローラはそのデータを受け取った後、2 回目の論理演算を行ない、新パリティ情報が求まりしだい、それをパリティノードに発送する。パリティノードが旧データを返送してから、新パリティが戻ってくるまでに掛かる平均時間は 5ms。

この 5ms を 1 回の論理演算時間とノード間の往復のデータ転送時間の和と考えると、データとパリティが両方ともヒットしたケースの同じ計測時間 3 の平均 9ms ＝”2 回の論理演算時間とノード間の往復のデータ転送時間の和” とから、1 回辺りの論理演算に約 4ms、ノード間のデータ転送に約 0.5ms かかる事が予想される。

話は戻って、この装置ではディスク新パリティを書き込むのに 5ms 掛かっている。新パリティがノードの届くまが 5ms。キャッシュへの書き込みは、瞬時にして行なわれるから、ほとんど同時に作業が終了する。

システム全体の総アクセス時間は 31〜32ms。上で見てきた処理に掛かった時間の和が約 28ms。その他の処理に 3〜4ms 掛かっているが、他の処理と比較してその値は妥当と言える。

表 5.5: 分散キャッシュ：データがヒット、パリティがミスした場合

分散キャッシュ：データがミス、パリティがヒットした場合

今度は前とは逆に、データがミスしてパリティ情報がヒットした場合を示す。表5.6に各動作に費やした時間を示す。図5.10に測定した処理の時間的位置を示す。

最初にキャッシュ探索までに掛かる平均時間は、前と同じくフルアソシアティブ方式で、ヒットした側で1ms、ミスした側で2ms費やしている。ダイレクトマップ方式も同様1ms未満で済んでいる。

キャッシュがヒットした後。旧パリティ情報はホストに送られ1回目の論理演算が施される。しかし、データノード側のディスクアクセス（旧データの読み込み）が終了してホストに旧データが送り返されるまで、2回目の論理演算は開始でいないから、その間パリティノード側は待ち続ける事になる。そのため、計測値3「旧パリティを返送した後、新パリティが戻ってくるまでの時間」は、27から28msと、これ以外のケースより長くなっている。ただし、その値自体は、" キャッシュミス側とキャッシュヒット側との間の探索時間の差と、旧データの読み込み時間と、2回のデータ転送時間と、論理演算時間" の和であるところの値・29msに近いので問題はない。

さて、データ側のデステージデータの書き込みに掛かる時間は、5ms。先ほどと同じく、2回目の論理演算に掛かる時間を5msと置けば、この場合の処理もほぼ同時に終る事になる。

総アクセス時間は、30ms〜32ms。この値は、データ側がヒットしてパリティ側がミスした場合と同じ位。この値自体は、分散キャッシュのデータとパリティが共に同時ヒットする場合に比べ、約2.5倍ほど大きい。しかし、キャッシュなしのRAID方式の38msと比べると、15%程短縮、2重キャッシュのデータミス時のアクセス時間と比べると15〜30%程短縮している。このことから、分散キャッシュは片側のみの不完全なヒットでも高速性が得られる事がわかる。

表 5.6: 分散キャッシュ：データがミス、パリティがヒットした場合

分散キャッシュ：両方ミスした場合

最後に、データとパリティ情報、どちらもミスしてしまった場合。表5.7に各動作に費やした時間を示す。図5.11に測定した処理の時間的位置を示す。

キャッシュ探索を始めてから終るまでに掛かる平均時間は、フルアソシアティブ方式で2ms。ダイレクトマップ方式で1ms未満。分散キャッシュでミスした場合としては妥当な値である。

この後、両方のディスクで読み込みが行なわれる。読み込みに掛かる時間は、これまた妥当な22ms。

読み込みが終了したら、旧情報をアレイコントローラに送って論理演算を2回行なう。ここまでは、キャッシュ探索以外はキャッシュ未搭載のRAID方式と等しく、この後データノード側がデータをデステージする所までは2重キャッシュ方式と、それぞれ良く似ている。分散キャッシュはさらに、パリティディスク側の書き込み作業も同時に行なう。ゆえに、アレイコントローラが旧パリティを受け取ってから、論理演算を終えて、新パリティをパリティノードに送るまでの時間（8ms）の間に、ディスクへの書き込み作業(5ms)は終了してしまう。

従って、ホストがデータを送ってからアクセス終了を受信するまでに費やすアクセス総計時間は、キャッシュ探索時間が短いダイレクトアクセス方式で36ms、探索に時間がかかるフルアソシアティブ方式で38ms掛かった。つまり、分散キャッシュではミス時でも（キャッシュ探索時間を抑えれば）、キャッシュ未搭載のRAIDより少ないアクセス時間で済む事がわかる。

表 5.7: 分散キャッシュ：両方ミスした場合

各方式のヒット or ミス時の処理時間の比較

表5.1〜5.7を基に、分散キャッシュ方式、2 重キャッシュ方式、キャッシュ未搭載方式を総じて比較したキャッシュ総容量 2000 ブロックの時のヒット時、ミス時の各作業に掛かるアクセス時間を図 5.12と図 5.13に示す。なお分散キャッシュでは、データノード側とパリティノード側は、並行して作業が行なわれるが、この図ではパリティノード側の情報のみを載せている。棒グラフの終端である所の処理の終了時間自体は、分散キャッシュの終了時間そのものと一致している。

フルアソシアティブ方式とダイレクトマップ方式の差異は、キャッシュ探索時間にある。フルアソシアティブ方式での分散キャッシュに比べて 2 重キャッシュのキャッシュ探索時間が大きいのは、1 回に探索するキャッシュに容量が、2 重キャッシュが「総キャッシュ容量/2」なのに対し、分散キャッシュは「総キャッシュ容量/6」で済むためである。なお、2 重キャッシュ方式、分散キャッシュ方式共にキャッシュがヒットした場合の探索時間は、キャッシュをすべて見なければならないミス時の大体半分になっている。

キャッシュ探索時間以外を比較した時、2 重キャッシュ方式がミスした場合とキャッシュ未搭載方式の処理時間は等しい。それに対して、分散キャッシュはデータとパリティ情報が両方ミスした場合でも、キャッシュ未搭載方式より、短い処理時間で終了している。これは先に述べたように分散キャッシュではデータの書き込みと論理演算処理を平行して行なえるためである。また、分散キャッシュではどちらか一方のみヒットした場合でも、さらに処理時間が短縮されている。これは 2 回行なう必要のある論理演算処理の 1 つをミスした情報の読み込みの間に行なえるためである。

分散キャッシュ方式のデータとパリティ情報が共にヒットした場合の処理が終了するまでの時間は、キャッシュ未搭載方式の 3 分の 1 までに短縮された。しかし、2 重キャッシュ方式のヒット時と比べればその時間は大きい。この事から、ヒット率が大きい状況では2 重キャッシュ方式の方がアクセス時間が短縮されると考えられる。

図 5.13: ヒット or ミス時の 1 回の処理時間（ダイレクトマップ方式）

図 5.14: 書き込みに掛かる平均アクセス時間

このグラフから、分散キャッシュ方式が、今回の実験の範囲内ではキャッシュ未搭載の RAID レベル 5 モデルに比べて、書き込み性能で常に勝っている事がわかる。

ダイレクトマップ方式は、キャッシュ容量が増えるに従って、さらなる時間の減少が見られた。フルアソシアティブ方式は、キャッシュ容量が増えても掛かる時間に変化がない

ように見える。これは、キャッシュ容量の増加によるキャッシュ探索時間の増加と、ヒット率上昇に伴うアクセス時間の減少がつりあってしまったためだと思われる。

本実験モデルでは、ソフトウエェア的にフルアソシアティブを実現しているため、キャッシュ探索に時間が掛かってしまっている。これさらに高速化出来れば、ヒット率の高さと合間って、ダイレクトマップ方式以上のアクセス時間の減少化が期待できるだろう。

また、この図では、キャッシュ容量が少ない時でも、低いアクセス時間を維持している。これはヒット率は小さくなるものの、キャッシュ探索時間も同時に少なくなるためである。前出の「データ・パリティ共にミスした場合」のキャッシュ探索時間が 0 に近くなったケースと考えれば良い。

2 重キャッシュと比べた場合、この範囲の実験では分散キャッシュは、2 重キャッシュより常に掛かる時間が短かった。ダイレクトマップ方式では、キャッシュ容量の増加と共にアクセス時間は減少していったが、分散キャッシュの同じダイレクトマップ方式との差はあまり縮まらなかった。フルアソシアティブ方式では、2 重キャッシュはキャッシュ容量の増大と共に、かかるアクセス時間も上がっていった。2 重キャッシュ方式は、キャッシュ探索時間の性能が分散キャッシュ方式の 3 倍はかかるので、その伸びもまた大きいかった

また、キャッシュ容量が 0 付近の時は、2 重キャッシュはキャッシュなしの RAID より若干時間が掛かっている。これは前にも述べたが、2 重キャッシュのミス時のアクセス動作は “キャッシュ探索時間＋キャッシュ未搭載 RAID5” なので、入力データがほとんどミスするだろうキャッシュの低容量時には、キャッシュ探索時間の若干の差が表に出てきているためであろう。

5.2 データの読み込み

この節では、実験システムの読み込みの際の性能評価を行なう。

5.2.1 ヒット率の比較

図 5.15は、2 重キャッシュと分散キャッシュ付きの RAID レベル 5 のフルアソシアティブ方式とダイレクトマップ方式のヒット率を比較したものである。

分散キャッシュの場合は、読み込み操作にパリティアクセスは必要ないので、データのヒット率が重要になる。ダイレクトマップ方式では、書き込みではデータとパリティに同

図 5.15: 分散キャッシュと 2 重キャッシュの読み込み時のヒット率

5.2.2 アクセス速度の比較

各方式での無故障時の読み込み操作は、パリティ演算がないので、単なるデータへのアクセスを行なうだけである。これは 2 重キャッシュも分散キャッシュも基本的に同じである。キャッシュの探索を行なって、ヒットしたらそのままキャッシュから読み込み、ヒットしなかったらディスクへ読み込みに行く。キャッシュは write-in(write-back) 方式なので、読み込みデータのキャッシュへの格納は行なわない。

キャッシュ総容量 2000 ブロックの時のヒット時、ミス時の各作業に掛かるアクセス時間を表 5.8 と図 5.16 に示す。

分散キャッシュ方式では、キャッシュ探索に 0〜2ms 以下、ディスクへのデータ読み込みに 22ms 掛かっている。総アクセス合計では、ミス時に 27ms、ヒット時に 3ms かかっている。書き込み時と比較した場合、ミス時で書き込み時より 15〜34% 程短縮、ヒット時は 75〜85%程短縮している。ミス時の書き込みに比べ、ミス時の読み込みの総アクセス時間がそれほど減少していないのは、この実験システムではディスクドライブへの書き込み処理がバッファのおかげで速くなっているのに比べ、ディスクドライブへの読み込み処理が遅いためである。

キャッシュ未搭載の RAID と比較した場合、フルアソシアティブ方式の分散キャッシュがミスした時、かかる時間はキャッシュ探索の分だけ遅くなっている。ダイレクトマップ方式のミス時の処理時間は、キャッシュ未搭載 RAID とほぼ同じであった。

2 重キャッシュ方式と、分散キャッシュ方式の処理時間を比較した場合。ダイレクトマップ方式では、ヒット時に若干の差が生じた。これは、2 重キャッシュ方式が、ホストノード上でキャッシュ探索を行なうのに対し、分散ノードはデータノードまで行ってキャッシュの探索を行なうからである。そのため、ノード間のデータ受渡しの分、分散キャッシュの方が遅くなっている。データがミスした場合は、ディスクまで読み込みに行かなければならないにで、同程度になるものと思われる。

フルアソシアティブ方式では、ミス時は分散キャッシュより 2 重キャッシュの方が 3ms 程度遅い。ただし構造的には、これはキャッシュ探査の時間に掛かる差と見る事が出来る。ヒット時は、分散ノードはディスクノードまでキャッシュを見に行く分ロスが生じるが、今回は、2 重キャッシュもキャッシュ探索に時間が掛かる分、結局両者の差は差引 0 になっている。

分散キャッシュ方式、2 重キャッシュ方式、キャッシュ未搭載方式を総じて比較した場合、キャッシュ探索時間を無視すればミス時の総アクセス時間はどれも差がないと言える。ヒット時は 2 重キャッシュ方式が極めて速く、分散キャッシュ方式はミス時の 1 割程度になっている。

5.2.3 実稼働性能の評価

最後に、5.1.3 節で行なった運動性能の評価実験を読み込み操作に対して行なった、その結果を示す。

実験方式としては、書き込みの時と同様。読み書きランダムに出現する形で各 2 万 5 千

表 5.8: ディスク読み込み時の各方式の作業時間

図 5.16: 読み込み:ヒット or ミス時の 1 回の処理時間

図 5.17: 読み込みに掛かる平均アクセス時間

分散キャッシュ方式の読み込み性能は、総キャッシュ容量が少ない時は、キャッシュ未搭載に RAID より、若干低い。これはキャッシュ探索に掛かる時間分だと思われる。しかし、ダイレクトマップ方式では、キャッシュ容量が増えるに従って、時間の減少が見られた。フルアソシアティブ方式は、キャッシュ容量が増えるとアクセス時間は緩やかに増加した。これもキャッシュ容量の増加によるキャッシュ探索時間の増加のためだと思われる。ヒット率上昇に伴うアクセス時間の減少も生じているのだろうが、この場合は探索時間の増加に負けているいうである。

分散キャッシュと 2 重キャッシュを比較した場合、ダイレクトマップ方式では、キャッシュ容量の増加と共に、両者とも緩やかに減少していった。

第 6 章

結論と考察

6.1 本研究の成果

本研究は、キャッシュに耐故障性を与えながら、RAID の問題点である書き込み速度を改善させる方法として、分散キャッシュ付き RAID の構造とその仕組みを考えた。

そしてディスクアクセスする事が出来る実験装置を作成した上で無故障時のヒット率と速度性能の測定を行なった。

その結果、分散キャッシュは書き込みにおいて、キャッシュ未搭載の RAID モデルと比べて、低い処理時間を実現できる事が判った。

同じく耐故障性を持った RAID 用ディスクキャッシュシステムである 2 重化したキャッシュ方式と比較した場合、ヒット率が低い場合では、分散キャッシュ方式の方が書き込みが高速である事が確認できた。

分散キャッシュ方式が他の方式と比べて比較的高速な書き込み操作を行なえるのは、データとパリティ情報がキャッシュで同時にヒットした場合、だけではなく、両方ミスした場合、あるいはどちらかの一方のみヒットした場合でも、短いアクセス時間で処理を完了する事が出来るからである。この要因は 2 つあり、1 つ目は、キャッシュのデータのディスクへの書き込みを新パリティ算出に必要な論理演算の終了を待たずに行なえるため。2 つ目は、片方のみがヒットした場合、ミスした側が古い情報を読み込んでいるうちに、論理演算の半分を済ませてしまうためである。

また、キャッシュ方法としては、フルアソシアティブ方式が、分散キャッシュに高いヒット率を与える事が判明した。しかし、今回の実験装置では、あまり良い結果が得られな

かった。これは、今回はキャッシュをソフトウェア的に実装したために、キャッシュ探索に速度的な問題を残してしまったからである、この点に関しては、より効率的なキャッシュ探索方法を用いればさらなる高性能化が期待できるものと思われる。

一方、読み込み性能においては、分散ディスクキャッシュは他の方式との大きな差を付ける事が出来なかった。これは、分散キャッシュ方式の処理の平行化が、単純なデータ読み込み作業に関しては作用しないためである。

6.2 今後の課題

今回の実験では、2 重キャッシュ方式の平均書き込み時間は分散キャッシュ方式の平均書き込み時間よりも大きかった。しかし、データがヒットした場合の 2 重キャッシュ方式のアクセス性能は、分散キャッシュ方式よりも良い。そのため、ヒット率が向上すれば、それに伴い平均の書き込み時間も向上し、ある時点で分散キャッシュの処理時を追い抜く事になると予想される。その条件を明らかにする事は 1 つの課題である。

もう 1 つの課題は、今回の実験では無故障時のシステム性能の測定を行なったが、このシステムは耐故障性を持ったシステムであるので、ディスクあるいはキャッシュでの障害発生時の速度性能の変化を調べる必要があると思われる。

その他の検討課題としては、ディスクへの書き込み処理が、論理演算処理よりも掛かるような場合に備える必要がある。1 つの方法としては、キャッシュからディスクへのデータのデステージ処理をホストからの命令とは非同期に予め行なう事が考えられる。

謝辞

本研究の全過程を通じ、直接丁寧な御指導、御鞭撻を賜わった北陸先端科学技術大学院大学情報科学研究科計算機アーキテクチャ講座横田治夫助教授に深謝致します。

本研究を行うにあたり、適切な御指導、御助言を頂いた日比野靖教授に心から御礼申し上げます。

研究室の杉野栄二氏、宮崎純氏、味松康行氏、麦飛氏、岩佐正道氏、松本英樹氏、また日比野、横田両研究室の皆様には種々の面でお世話になりました。以上の皆様には深く感謝の意を表します。

参考文献

[1] Menon.J and Cortny.J, "The architecture of fault-tolerant cached RAID controller", *IEEE.NEW York,76-8,* 1993.

[2] Peter.M.Chen etal, "RAID:High-Performance,Reliable Secondary Storage", *ACM computing Surveys Vol.26 No2* , June,1994.

[3] JAI.MENON, "Performance of RAID5 Disk Arrays with Read and Write Caching", *Distributed and Parallel Database,2,261-293,* 1994.

[4] 横田治夫, "RAID のネットワーク上への展開と信頼性向上", 信学技報 CPSY 93-11, FTS 93-11, ICD 93-11, 電子情報通信学会 (1993).

[5] 友永誠史, "並列処理環境における二次記憶システムの信頼性における研究", 修士論文 (1994).

[6] 岩佐正道, "二次元トーラス結合におけるリンク故障及びノード故障の取り扱いに関する研究", 修士論文 (1996).

[7] 齊藤忠夫, 発田 弘, "高性能コンピュータアーキテクチャ" 丸善株式会社, 1989.

[8] 石畑 清, "アルゴリズムとデータ構造", 岩波書店, 1992.