博 士 論 文

ネットワーク結合型並列ディスクシステムに関する研究

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

味松 康行

1998 年 1 月 16 日

要旨

計算機システム全体の処理速度を上げるためには、プロセッサやメモリの高速化のみならず二次記憶としてのディスクシステムの性能向上が不可欠である。また、その大容量化の際にはシステムの信頼性を保つことが重要である。

RAID として知られるディスクシステムは、複数のディスクを並列に動作させることにより性能の向上を図り、冗長情報を利用することでシステムの信頼性低下を防ぐ。しかし、RAID では多くのディスクをバスで結合するため、ディスク台数が多い構成ではバスが通信のボトルネックとなることが考えられる。また、信頼性に関しても冗長情報を管理する単位グループ内での複数のディスク故障に対する考慮が必要である。このように、大規模な構成を考慮した場合には、RAID は十分な性能や信頼性を提供できないと考えられる。

それに対し、RAID を改善したアーキテクチャの一つとして Data-Reconstruction networks (DR-net) が提案されている。ネットワークによるディスク間の結合、独立に動作する任意個の外部インタフェースの利用、任意の 2 つのディスク故障のマスクという 3 つの特徴により、特に大規模な構成においては DR-net は RAID よりも性能および信頼性が改善されると期待される。しかし、DR-net の性能や信頼性の解析、評価はこれまで十分ではなかった。

そこで、本研究では DR-net の信頼性および性能の評価を行ない、上記の特徴により、性能や信頼性が向上することを検証する。また、いくつかの異なる動作方式についての考察や書き込み性能に関する DR-net の問題の指摘と解決により、DR-net が大規模な二次記憶として実用的であることを示す。

信頼性に関しては、MTTF による RAID と DR-net の比較および一般的な構成規模でのマスク可能故障数の比較により DR-net が RAID レベル 3 〜 5 よりも高信頼なシステムであることなどを示す。性能面では、パリティの分散方式やデータの再構築戦略などの動作方式と性能の関係について小型実験システムを用いた測定結果を示し、それぞれの特性を明らかにする。さらに、シミュレーション結果からディスクノード数や通信バンド幅の性能への影響を明らかにし、DR-net が大規模構成に適していることを示す。また、DR-net の問題点である書き込み性能について考察し、ディスクキャッシュやログを利用した書き込みなどの解決策を検討する。

本研究の結果、大規模構成における DR-net の RAID に対する優位性が明らかとなり、問題点の解決と併せて、DR-net による高性能、高信頼な大規模二次記憶システムの実現が期待できる。

目 次

第 1 章

序論

近年、プロセッサの単体性能の向上にともない大量のデータを高速に処理することが期待されているが、計算機システム全体の処理速度を上げるためには、二次記憶として用いられるディスクシステムの性能向上が不可欠である。また、大容量化の際にはシステムの信頼性を保つことが必要である。しかし、プロセッサやメモリのような半導体製品とは異なり、ディスクシステムには機械的な動作が伴うため、単体ディスクにおける性能および信頼性の大幅な向上は困難である。

Patterson らが提案した Redundant Arrays of Inexpensive Disks(RAID)[26] は、並列に動作可能な複数のディスクを 1 つのシステムに統合することにより性能の向上を図り、またシステム内に冗長な情報を持つことによりシステムの信頼性低下を防ぐ。今日では商品として発表された RAID システムも多く、またさらなる改善を目指した研究も盛んに行なわれている。

RAID を改善したアーキテクチャの 1 つとして Data-Reconstruction networks(DR-net) がある。DR-net では RAID と異なり、ディスクをバスではなくネットワークで結合する。これにより通信ボトルネックを解消することができる。また、独立に動作する任意個の外部インタフェースを持つことができるため、インタフェース (コントローラ) での負荷の集中を緩和することができる。信頼性に関しても、新たな冗長情報の利用を導入することで任意の 2 つのディスクが故障してもデータが失われない。これらの 3 つの特徴により、特に大規模な構成においては DR-net は RAID よりも性能および信頼性が改善されると期待されている。

しかし、DR-net の上記のような特徴に対する評価はこれまで十分でなかった。上記の特

徴が実際に性能や信頼性に反映されることを示すためには DR-net の個々の特徴の解析を行ない、RAID と比較した評価も必要である。

また、これまでの研究は主に DR-net の長所について議論されてきたが、DR-net の問題点についても考察し、それに対する解決策を示すことも重要である。本研究では、DR-net に関して次の 4 つのことについて考察する。

(1) 信頼性

(3) さまざまな動作方式と性能の関係

(2) 構成規模や通信バンド幅の性能への影響

(4) 問題点の指摘とその解決

(1) では、RAID には見られない互いに重なり合う 2 種類のパリティグループの利用による信頼性向上について、特定の規模の MTTF を RAID と比較することにより検証する。また、冗長な情報量を低減した構成や一般的な規模の構成での信頼性についても言及する。

(2) では、DR-net におけるパリティの分散方法や、故障ディスク内のデータの再構築の方法の違いが性能に与える影響について、小規模な実験システムを用いた評価結果を示し、各方式の特性について考察する。

(3) では、DR-net の構成上の特徴である、内部ネットワークと複数の外部インタフェースの利用の効果をシミュレーションにより検証する。ディスクノード数、インタフェースノード数、通信バンド幅などの変化とスループットやレスポンスタイムの関係について、階層バスを用いる RAID システムとともに評価する。

(4) では、DR-net ではデータの読み出しに比べてデータの更新の性能が低いことに関して、キャッシュメモリやログを用いる書き込みの高速化手法の DR-net への導入を検討し、その効果を検証する。

以上の 4 つにより、DR-net が大規模構成に適したシステムであり、高性能/高信頼な二次記憶として実用的であることを示す。本研究の結果、RAID では不十分な性能しか得られない大規模な構成においても、高性能で信頼性の高い二次記憶の実現が期待できる。このことは現在の I/O バウンドのアプリケーションの高速化だけでなく、より大量のデータを扱う新たなアプリケーションやシステムの開発にも寄与すると考えられる。

以下に、本論文の構成を述べる。第 2章では並列ディスクシステムである RAID について概説し、RAID の特徴および問題点について述べる。第 3章 では本研究の対象である DR-net

の基本的な構成、動作、特徴について述べる。第 4章では DR-net の信頼性について評価する。第 5章では、いくつかの DR-net の動作方式について比較し、それぞれの特性を明らかにする。第 6章では、大規模な構成における内部ネットワークの有効性や複数のインタフェースを持つ効果などについての実験結果を示す。第 7章では、DR-net の書き込み性能の向上について議論する。最後に第 8章で結論を述べる。

第 2 章

RAID

RAID(Redandunt Arrays of Inexpensive Disks)[26] は、複数のディスクを 1 つに統合した二次記憶システムである。多数のディスクを用いることにより大容量を、並列アクセスにより高性能を、冗長情報の利用により高信頼性を達成する。1980 年代後半から盛んに研究され、今日では商品として発表された RAID システムも数多い。本章ではこれまでに研究されてきた RAID の特徴および問題点などについて述べる。

2.1 構成

RAID の構成は冗長情報の格納方式やデータ分割の粒度により、7 つのレベルに分類される [5]。各レベルの特徴を簡単にまとめる。

レベル 0 冗長情報を持たず、いかなる単一ディスク故障でもデータが失われる。しかし、冗長情報更新のオーバヘッドがないため、もっとも高い書き込み性能を持つ。読み出し性能は次に述べるレベル 1 よりも劣る。

レベル 1 各ディスクを二重化する。同じデータを 2 つのディスクのどちらからでも読み出せるため、高い読み出し性能を持つ。各データが 2 重化されているため、1 つのディスクが故障してもデータは失われない。

レベル 2 データをビット単位で全ディスクに分割し、ハミング符号を用いて冗長情報を付加する。記憶効率が悪く、ほとんど利用されない。

レベル 3 レベル 2 と同様にデータはビット単位で分割され、冗長情報として単一のパリティを用いる。1 回のアクセスに全てのディスクが使われるため同時に複数のアクセスを処理できない。バンド幅は広いが、I/O レートは低い。

レベル 4 データをストライピングユニットと呼ばれるブロック単位で分割して、複数のディスクに格納する。小さなデータへのアクセスには少数のディスク集合のみが使用されるため、同時に複数の要求を処理することができる。冗長情報として、いくつかのディスクブロックのパリティを特定のディスクに保持しておく。レベル 4 では、パリティを特定の単一ディスクに保持しているため、データ更新の際にそのディスクに負荷が集中し、性能のボトルネックになる傾向がある。

レベル 5 レベル 4 のボトルネックを解消するため、パリティを全てのディスクで分散保持する。このことにより更新処理での負荷が分散される。また、全ディスク数を N とすると、レベル 4 ではデータは $N - 1$ 個のディスクで保持されていたが、この構成では全てのディスクに分散されることになり、同じデータ量に対してより並列にアクセスすることができる。

レベル 6 パリティではなく、Reed-Solomon 符号を用いて冗長情報を付加する。レベル 1 から 5 までは単一ディスク故障に対応しているが、この構成は 2 つのディスク故障に対応できるため、大容量の構成でも信頼性を保つことができる。

これらのうち、レベル 5 は小さなデータの読み出し、大きなデータの読み出し、大きなデータの書き込みにおいて高い性能を持つため、レベル 5 を基本とする構成についての研究が盛んに行なわれている。

2.2 RAID レベル 5

図 2.1に、5 台のディスクからなる RAID レベル 5 の構成を示す。5 台のディスクがそれぞれバスに接続され、ユーザは RAID コントローラをインタフェースとして外部からアクセスする。

図 2.2は内部でのデータの配置を表す。図のデータブロックにはデータが、パリティブロックには他のディスクのデータブロックに保持されるデータのパリティが保持される。

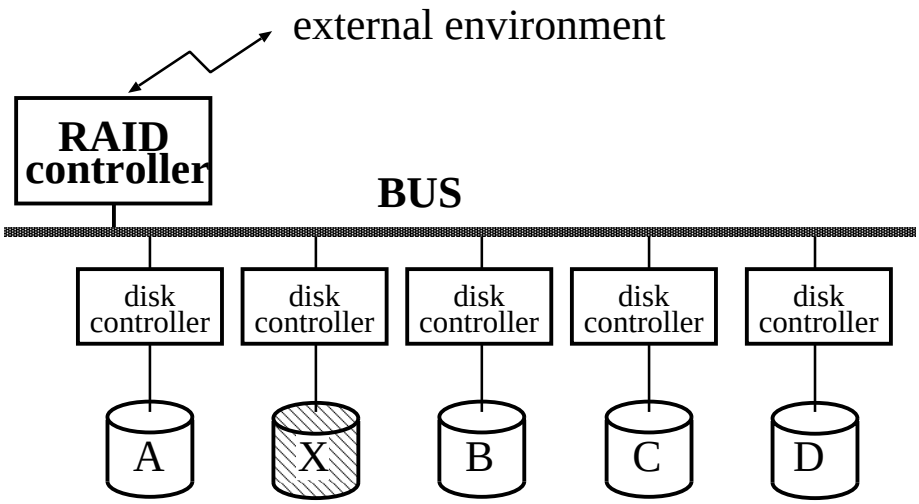

図 2.1: RAID レベル 5 の構成

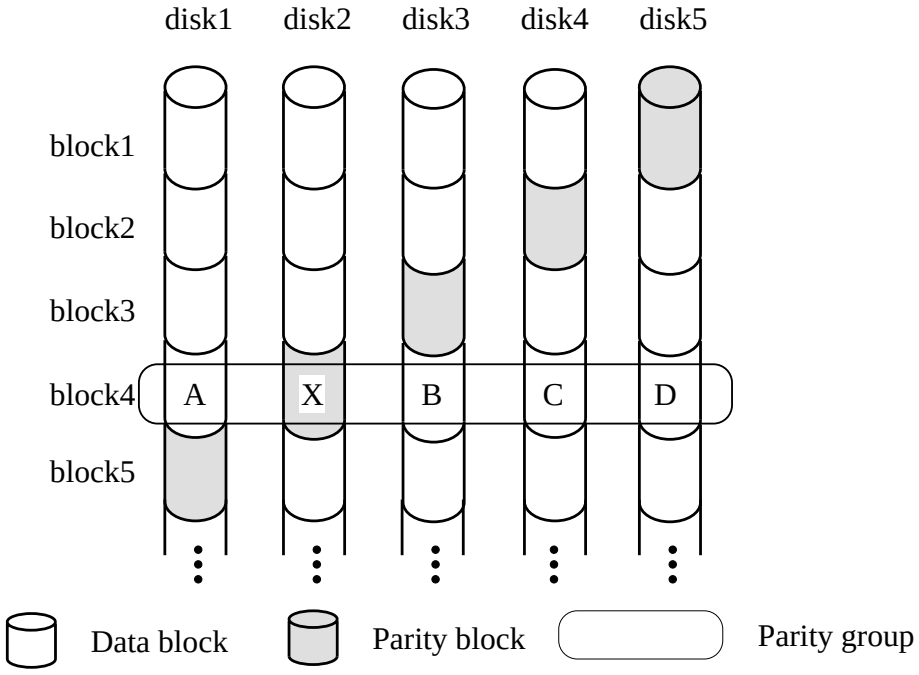

図 2.2: RAID レベル 5 でのデータの格納

図 2.2のブロック A, B, C, D, X には

$$X = A \textbf{ xor } B \textbf{ xor } C \textbf{ xor } D \tag{2.1}$$

の関係がある。このような、1 つのパリティブロックとそれを計算するために必要なデータブロックの集まりをパリティグループと呼ぶ。1 つのパリティグループにシステム内の全てのディスクが含まれる必要はないが、全てのブロックは必ず 1 つのパリティグループに属す。

書き込みでは常にパリティの更新が伴う。例として図 2.2のブロック A を書き換えるときは、新しいパリティは次の式で計算される。

$$X_{new} = A_{old} \textbf{ xor } A_{new} \textbf{ xor } X_{old} \tag{2.2}$$

書き込みには古いデータの読み出し、新しいデータの書き込み、古いパリティの読み出し、新しいパリティの書き込みの合計 4 回のディスクアクセスおよびパリティ演算が必要となる。これは read-modify-write として知られる高コストな処理である [5]。パリティブロックを保持するディスク (パリティディスク) はブロック毎に異なり、全てのディスクが均等にパリティブロックを保持する。

ディスク故障が発生した場合は、式 (2.1) を逆算することでデータを再構築できる。図 2.2の例ではディスク 1 が故障した場合のブロック A の内容は

$$A = X \textbf{ xor } B \textbf{ xor } C \textbf{ xor } D \tag{ 23}$$

で再構築できる。

2.3 RAID レベル 5 の問題点

複数のディスクを統合し、また、冗長情報を利用することにより RAID は高性能、高信頼性を提供する。しかし、プロセッサの処理性能の向上や並列処理の発達による大量のデータ処理、あるいは、動画を含む画像や音声などのマルチメディアアプリケーションに対しては、より大容量で高性能な二次記憶が求められる。さらに、大容量化に際しては信頼性を保つことも重要である。このような要求に答えようとするとき、RAID のシステム構成が問題となる。

2.3.1 単一コントローラとバスによる性能のボトルネック

容量を増やすためには、個々のディスクの容量を引き上げることと、ディスクの台数を増やすことの 2 つが考えられる。しかし、性能の向上を併せて考慮する場合には、ディスクの台数を増やし、アクセスの並列度を上げることが望ましい。

しかし、ディスク台数を増やして性能を向上させるときに、図 2.1のような構成では単一の外部インタフェース (RAID コントローラ)、および多数のディスクを結合するバスが性能面でボトルネックになると考えられる。特に、将来的にディスク台数が数千台にものぼるような大規模システムでは、スループットの限界はバスとインタフェースのバンド幅で決定され、ディスク台数の増加に見合う性能の向上は望めない。

また、現在既に数百 Mbits/sec の転送レートを持つような単体ディスクの研究が行なわれている。マルチプロセッサマシン用の内部バスのバンド幅でも 1GBytes/sec 程度であることを考えると、このような高性能ディスクの使用を前提とした場合には、ボトルネックはいっそう顕著に影響するものと思われる。

さらに、RAID システムでは通常のユーザからのディスクアクセス以外にもパリティの更新処理や、故障が起きた場合のデータ再構築処理でもバスが使用されること、高速で距離の長いバスの実現が難しいことなどから、バスを用いたディスクアレイにはスケーラビリティに限りがあり、単純なバス結合によるディスクアレイは大規模構成には適さないと言える。

2.3.2 大規模構成における信頼性の低下

ディスク台数の増加は、システムの信頼性にも影響を及ぼす。通常、他の条件が同じであれば、RAID システムの MTBF(Mean Time Between Failures) はディスク台数に反比例して低下する [3]。

そのため、大規模システムの信頼性を維持するためにはより多くの冗長情報を利用することにより、マスクできる故障ディスクの台数を増やすことが必要となる。パリティグループ内の単一ディスク故障をマスクできるだけのシステムでは、十分な信頼性が得られないことが予想される [3][10]。この問題を改善した例として RAID レベル 6 や EVENODD[1][2] が存在する。これらは、1 つのパリティグループ内の 2 つのディスク故障をマスクできる。

このように、今後、大規模なシステム構成を考える際にはパリティグループ内での単一故障ではなく、グループ内での複数故障にも対応することを考慮した議論が必要である。

第 3 章

データ再構築ネット (DR-net)

前章で見たように、RAID レベル 5 は信頼性および性能の両面で大規模構成には不十分である。データ再構築ネット (Data-Reconstruction networks:DR-net) は、そのような RAID の欠点を解消した高信頼並列ディスクシステムである [36]。

DR-net はディスクを相互にネットワークで結合し、通信を分散させることによりバスシステムで見られるようなボトルネックを解消している。また、複数の外部インタフェースを用いることによりコントローラの負荷分散と通信バンド幅の向上を実現する。このことにより、構成規模が大きくなってもディスク台数に応じた性能向上が望める。また、RAID と同様にパリティを冗長情報として用いるが、1 つのディスクが 2 つのパリティグループに属すことにより、パリティグループ内での多重ディスク故障に対しても耐故障性を有する。

本章では本研究の対象である DR-net の構成, 動作およびいくつかの動作方式について概説する。

3.1 DR-net の構成

DR-net は独立に動作する任意個の外部インタフェースノードと、ディスクにデータやパリティを格納してパリティグループを構成するディスクノードをそれぞれネットワークで結合する (図 3.1)。以下に、各構成要素について説明する。

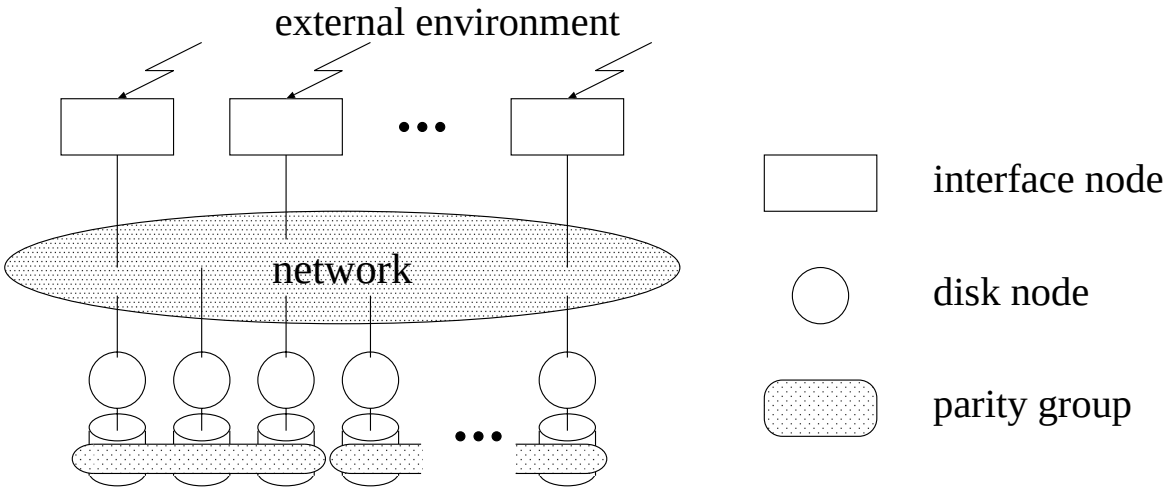

図 3.1: DR-net の概念図

3.1.1 ディスクノード

ディスクノードは、ディスクドライブとコントローラからなり、互いにネットワークで結合されている。ディスクにはデータやパリティが格納される。コントローラはドライブの制御, 通信, パリティ計算のための XOR 演算などを行なう。ディスクに対するアクセス要求を受信すると、コントローラが要求の内容に応じてディスクにアクセスする。また、書き込みなどのパリティ処理を伴う場合には、同じパリティグループに属すノード間で通信し、必要な処理を行なう。各ノードは独立に動作し、メッセージ交換で同期をとる。

パリティグループ内で、データを保持するノードをデータノード、またそのディスクをデータディスクと呼ぶ。パリティを保持するノードはパリティノード、そのディスクをパリティディスクと呼ぶ。

3.1.2 外部インタフェースノード

ホストコンピュータなどの外部環境とのインタフェースとして必要なノードは外部インタフェースノード、あるいは単にインタフェースノードと呼ばれる (図 3.2)。インタフェースノードはユーザからのアクセス要求を各ディスクノードに送信し、結果を受信する。

DR-net では複数のインタフェースノードを用いることが可能で、インタフェースにおける性能のボトルネックやインタフェースノードの故障によりデータにアクセスできなくな

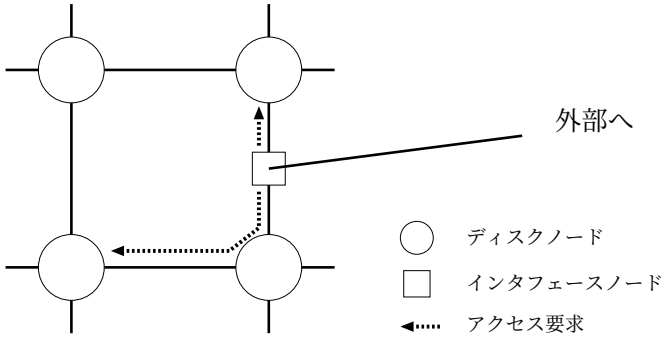

図 3.2: 外部インタフェースノード

ることを避けることができる。インタフェースノードが複数存在する場合には、それらは独立に動作することができる。インタフェースノードはネットワークの任意の位置に配置することができるが、通信の集中を避けるために、なるべくネットワーク内に分散して配置することが望ましい。インタフェースノードはディスクを持たず、パリティ計算などには関わらない。

3.1.3 パリティグループ

一つのパリティグループは、複数のデータノードと一つのパリティノードからなる。RAIDの場合と同様に、複数のデータノードに保持されるデータのパリティが1つのパリティノードで保持される。パリティ更新やデータ再構築の際には、各パリティグループ内で通信を行なう必要がある。通信を局所的に行なうためには、データノードはなるべくパリティノードの近傍に配置されることが望ましい。

DR-netには2種類のパリティグループがあり、それぞれFPG(First Parity Groups), SPG(Second Parity Groups) と呼ぶ。1 つのディスクノードは 1 つの FPG と 1 つの SPG に属し、同じ FPG(SPG) に属すディスクノードは互いに異なる SPG(FPG) に属す。この 2 つのパリティグループを併用することで、パリティグループ内の複数故障をマスクできる。FPG と SPG の具体的な利用法については後述する。

3.1.4 ネットワーク

ノード間を結合するネットワークのトポロジは各種考えられるが [34][36]、通信を効率良く行なうためにはネットワークの直径が小さく、また各パリティグループを局所的にマッピングできることが望ましい。

DR-net の内部ネットワークとして、ここでは、これまでに最も良く研究されている 2 次元トーラスを用いた構成について説明する。特に断らない限り以後は最小の構成単位である 5×5 の構成について説明する。

2 次元トーラスネットワーク上で、FPG はパリティノードを中心とする十字型 (図 3.3)、SPG はパリティノードを中心とする斜め十字型 (図 3.4) に配置される [34][36]。このようなパリティグループを用いる場合、ネットワーク全体にパリティグループを隙間なく配置するためには、トーラスネットワークの一辺は 5 の倍数でなければならない。

この構成ではデータディスク 4 台, パリティディスク 1 台の合計 5 台のディスクで 1 つのパリティグループを構成するが、1 つのパリティディスクは FPG, SPG のパリティを両方保持しなければならないからデータディスクの 2 倍の容量が必要である。したがって、データとパリティの容量比は 2:1 となる。この容量の不均衡は、3.3.1で説明するようにパリティを分散配置することで解消される。

3.2 DR-net の動作

外部から DR-net へのアクセスは、インタフェースノードを介してデータノードにアクセス要求パケットを送信し、データノードからの結果を受信することで行なわれる。

アクセスの単位としてはファイル単位あるいはディスクブロック単位が考えられるが、DR-net は特定のファイルシステムを必然的に包含するシステムではなく、さまざまなファイルシステムに記憶空間を提供する、より低レベルなシステムである。従って、以後特に断りがない限り、ファイルではなくディスクブロックをアクセスの単位とする。

以下で、インタフェースからディスクノードに対して読み出しまたは書き込みのアクセス要求が送られた場合の各ノードの処理を説明する [34][36]。

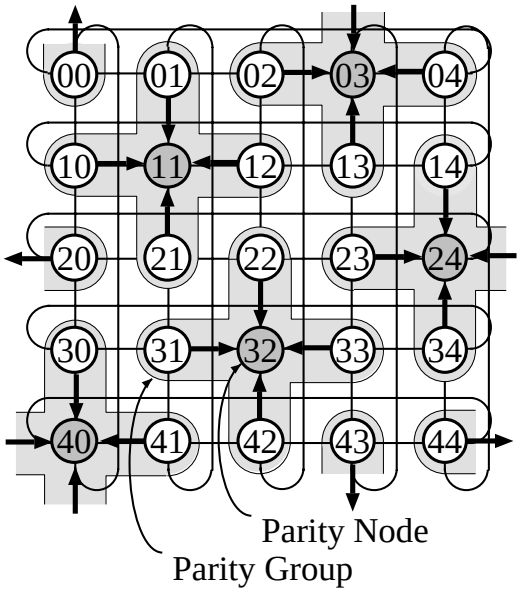

図 3.3: FPG(First Parity Groups) の構成

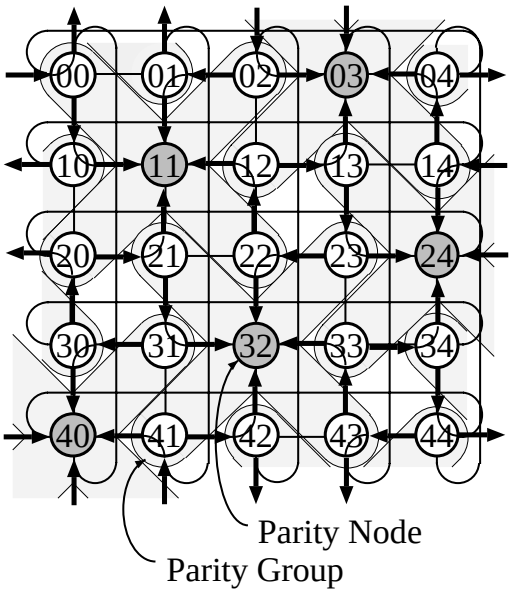

図 3.4: SPG(Second Parity Groups) の構成

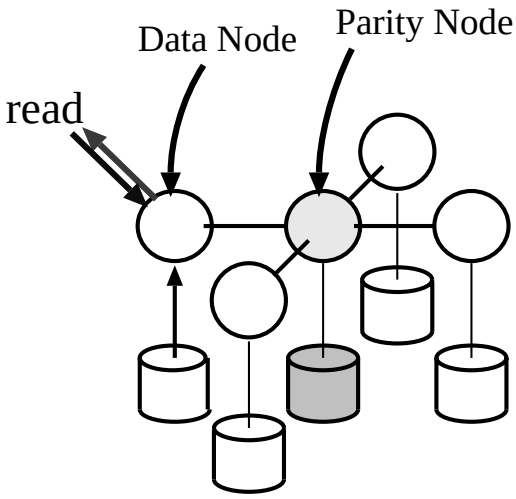

図 3.5: 読み出し (ディスク故障なし)

3.2.1 データディスクが故障していないとき

読み出し

読み出し要求を受信したデータノードのディスクが故障していなければ、内容を読み出して返信する。これは、通常のディスクアクセスと全く変わらない (図 3.5)。

書き込み

データを更新する際には、パリティの更新が伴う。データを書き込んだデータノードが属す FPG および SPG のパリティを両方とも更新しなければならない。

新しいパリティは、データディスクが故障していない場合には、RAID レベル 4 または 5 と同様に次式で算出される。

$$\text{新パリティ} = \text{旧パリティ} \ \textbf{xor}\ \text{旧データ} \ \textbf{xor}\ \text{新データ} \tag{3.1}$$

処理の流れは次のようになる (図 3.6)。

1. データノードはディスクから旧データを読み出す。
2. 新データを書き込み、同時に新旧データの差分パリティを計算し、結果を FPG および SPG のパリティノードに送る。

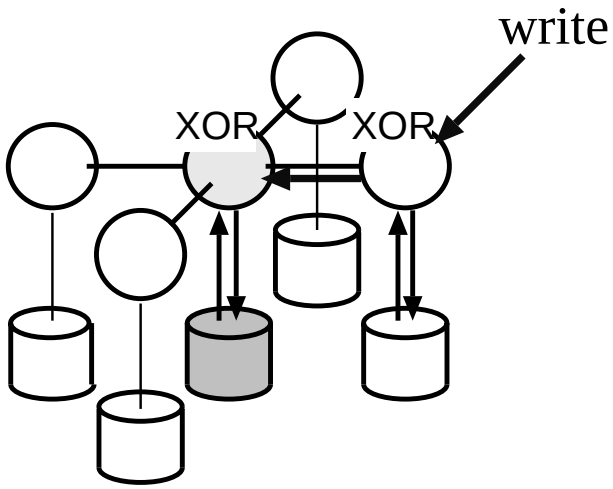

図 3.6: 書き込み (ディスク故障なし)

3. パリティノードは、ディスクから読み出した旧パリティと受信した差分パリティから新パリティを生成し、書き込む。

旧パリティの読み出しでパリティディスクの故障が検出された場合には、パリティの更新は行なわれなず、データディスクに新データが書き込まれるだけである。

3.2.2 データディスクが故障しているとき

読み出し

要求を受けたデータノードのディスクが故障してるときは、次のように処理する (図 3.7)。

1. アクセス要求を受け取ったデータノードが、ディスク故障を検出する。
2. パリティノードにデータの再構築を要求する。
3. パリティノードはパリティを読み出し、同時に同じパリティグループの他のデータノードに読み出し要求を出す。
4. パリティノードから読み出し要求を受けたデータノードは、データを読み出してパリティノードに返信する。

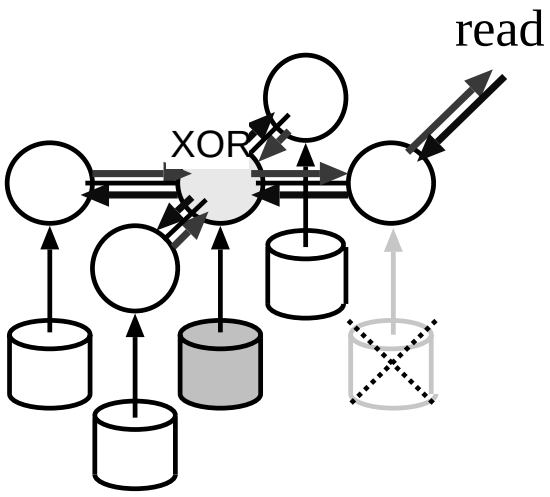

図 3.7: 読み出し (ディスク故障時)

5. パリティノードは、返信された全てのデータとパリティを XOR 演算し、結果を再構築結果としてデータノードに返信する。

6. データを受けとったデータノードは、インタフェースノードにデータを送る。

再構築を要求するパリティノードは、FPG あるいは SPG のどちらのパリティノードでもよい。

書き込み

故障ディスク内のデータの更新は、そのデータが属すパリティグループのパリティを更新することにより実現できる。パリティ計算の際、式 (3.1) のパリティ計算に必要な書き換え前の旧データは故障ディスク内にあるため、パリティは次式で計算される。

$$\text{新パリティ} = \text{新データ } \textbf{xor} \text{ データ A } \textbf{xor} \text{ データ B } \textbf{xor} \text{ データ C} \qquad (3.2)$$

ここで、データ A, B, C は更新するデータと同じパリティグループに属す 3 つのデータである。一連の処理は次のようになる (図 3.8)。

1. アクセス要求を受け取ったデータノードが、ディスク故障を検出する。

2. パリティノードに新データを送り、パリティの更新を要求する。

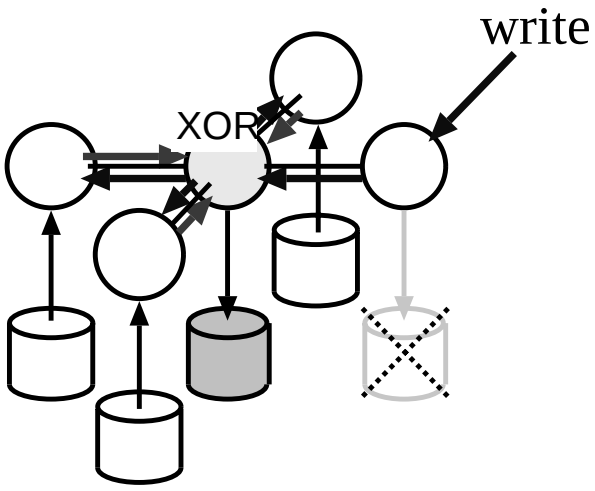

図 3.8: 書き込み (ディスク故障時)

3. パリティノードは同じパリティグループの他のデータノードに読み出し要求を出す。
4. パリティノードから読み出し要求を受けたデータノードは、データを読み出してパリティノードに返信する。
5. パリティノードは、返信された全てのデータと新データを XOR 演算し、結果を新パリティとしてディスクに書き込む。

パリティ項新要求は、FPG および SPG の 2 つのパリティノードに送らなければならない。

3.2.3 パリティグループ内の多重ディスク故障への対応

1 つのパリティグループ内で複数のディスク故障が発生した場合には、前述の 2 種類のパリティグループ (FPG, SPG) を併用することで対応できる [34][37]。同じパリティグループ内の 3 つのデータノードでディスク故障が発生している図 3.9の例で、ノード 31 のデータを読み出す場合を考える。

1. ノード 31 が属す FPG を用いてデータの再構築を試みる。再構築の際には、前節で述べたように同じ FPG に属す他の 3 つのデータノードのデータおよびパリティノードのパリティが必要である。

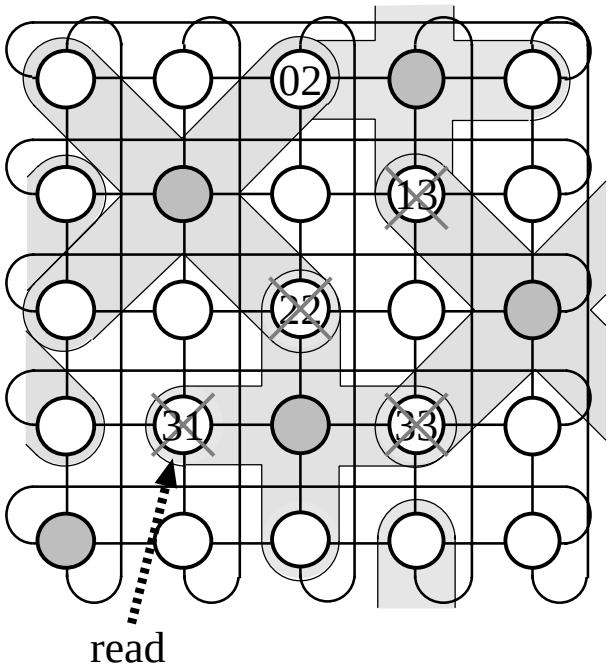

図 3.9: FPG, SPG を併用したデータ再構築

2. 必要なデータを保持する 2 つのデータノード 22, 33 で、ディスクが故障していることが検出される。

3. ノード 22, 33 のデータをそれらが属す SPG を用いて再構築する。

4. ノード 33 の再構築では、再構築に必要なノード 13 が故障しているので先にノード 13 の内容をその FPG を使って再構築する。

5. ノード 22, 33 の再構築結果を用いてノード 31 の内容を再構築する。

このように再構築に FPG と SPG を交互に使うことで、1 つのパリティグループ内に複数のディスク故障が存在する場合のデータ再構築が可能である。

再構築に必要なデータやパリティが 1 つでも得られない場合は、再構築は不可能であり故障ディスクのデータは失われる。故障ディスク数が多くなると再構築が不可能となる確率が高くなるが、同じ故障数でも故障が発生したノードの配置パタンにより再構築可能な場合と不可能な場合が存在する。5×5 の 2 次元トーラス構成では、いかなる 2 つのディスク故障でもデータは失われず、最大 9 個の故障でもデータが失われない場合がある [34]。DR-net の信頼性に関する詳しい議論は第 4章で述べる。

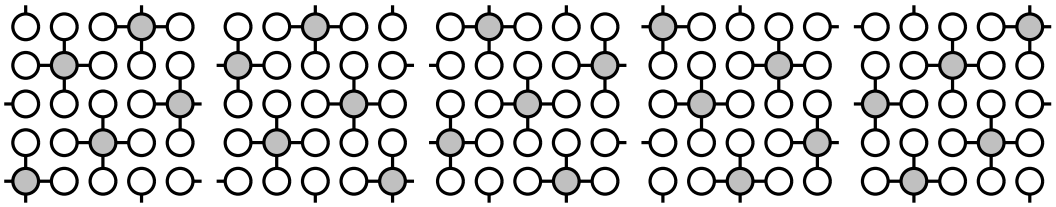

図 3.10: MPG:Moving Parity Groups(フェーズ毎のパリティグループの移動)

3.3 DR-net の動作方式

DR-net には、データの再構築を行なう戦略やパリティの配置方法の違いにより、いくつかの方式が存在する。

3.3.1 パリティ分散保持方式

パリティを保持するノードをこれまで述べてきたように特定のノードに固定してしまうと、パリティ更新処理の集中およびディスク容量の不均衡という問題が生じる。これらの問題を解決するパリティ分散保持方式として次の 2 つが提案されている [34][37]。

MPG

一つは、ネットワークの対称性を利用し、パリティグループの配置をフェーズ毎にずらす方法である (図 3.10)。5×5 の 2 次元トーラスネットワークの場合では、図 3.10のようにどのノードも 5 つのフェーズのいずれかでパリティノードとなる。フェーズは、アクセス要求毎に、アクセスするブロック、トラック、シリンダ等により切替える。このパリティ分散方式を MPG(Moving Parity Groups) と呼ぶ。この方式はフェーズによって一つのパリティグループに属すディスク群が異なるので、使用ノードを各パリティグループの近傍に限定した Declustered Parity[5][19][22] とも考えられる。

MPN

もう一つの方法は、パリティグループの配置は固定し各パリティグループ内でパリティノードの位置を移動させるものである (図 3.11)。この方式を MPN(Moving Parity Nodes) と

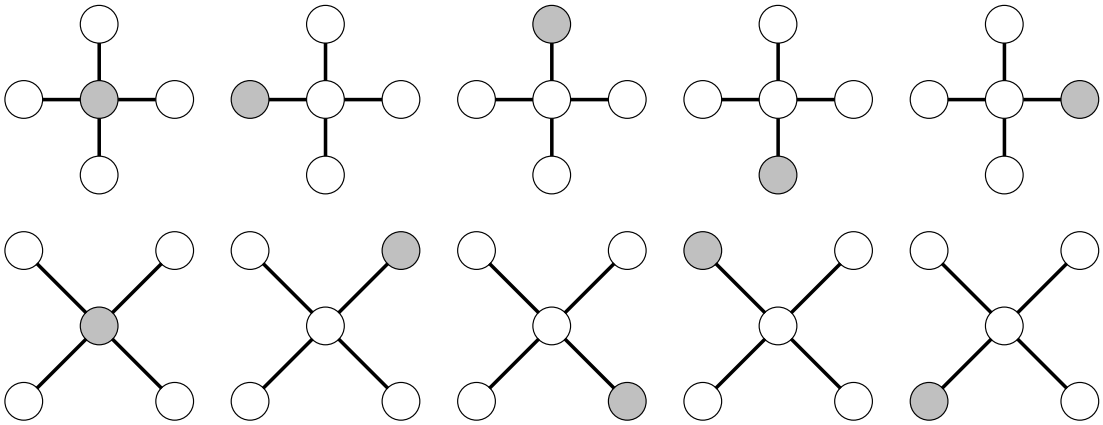

図 3.11: MPN:Moving Parity Nodes(フェーズ毎のパリティノードの移動)

呼ぶ。MPN は非対称なネットワークにも適用することができるが、各パリティグループ内でデータノードからパリティノードまでの平均通信距離が若干増大する欠点を持つ。

3.3.2 データ再構築戦略

故障ディスクのデータを再構築する場合、FPG と SPG のどちらでも利用できる。FPG を用いた再構築に失敗しても SPG を用いた再構築に成功する場合がある。また、その逆もあり得る。再構築のために利用できるパリティグループが 2 種類あることから、次の 2 つの再構築戦略 (LRS, ERS) が考えられる [40]。

LRS

LRS(Lazy Reconstruction Strategy) は、2 つのパリティグループを逐次的に用いる戦略である。まず 1 つのパリティグループ (FPG あるいは SPG) で再構築を試み、成功すればその結果を返す。失敗した場合にのみ、もう一方のパリティグループを用いた再構築を試みる。この戦略では最初の再構築に成功したときには、1 つのパリティグループだけを使うため、次の ERS に比べて無駄な処理が減る。しかし、最初の再構築に失敗した場合には、失敗が確定してからもう一方のパリティグループを用いた再構築を始めるため、再構築完了までにより長い時間を要する。

ERS

ERS(Eager Reconstruction Strategy) は、2 つのパリティグループを並列に用いる戦略である。FPG, SPG で同時に再構築処理を開始し、最初に返ってきた結果を利用するため、より速いレスポンスを期待できる。反面、最初に返ってきた再構築結果が成功であった場合、後から返される結果は利用されず捨てられてしまう。つまり、無駄な再構築を行なっていることになり不必要な処理が増加してしまう。

3.4 まとめ

DR-net の特徴をまとめると以下のようになる。

1. ノードは相互にネットワークで結合され、通信が分散されるため、RAID システムにおけるバスのような通信ボトルネックが存在しない。
2. 任意の数のインタフェースノードを設置できる。また、そのネットワーク内の位置も任意であるため、インタフェースの負荷と通信を分散させることができる。
3. FPG, SPG の 2 種類のパリティグループを用いることにより、いかなる 2 つのディスク故障が発生しても、データは失われない。また、パリティグループ内で 3 つ以上の故障が発生しても、データが失われるとは限らない。
4. MPG, MPN の 2 通りのパリティの分散保持方式がある。
5. 2 種類のパリティグループが存在することから、LRS, ERS の 2 通りのデータ再構築戦略が考えられる。

以降の各章では、これらの特徴を生かした性能および信頼性の向上について具体的に検証する。上記の 1, 2についてはシステムのノード数や通信バンド幅の性能への影響を第 6章で議論する。第 4章では、主に 3による信頼性の向上について考察する。5, 4のそれぞれの動作方式と性能の関係については第 5章で述べる。

第 4 章

DR-net の信頼性

本章では、DR-net の信頼性を評価し、DR-net の特徴の 1 つである複数のディスク故障への対応の効果を検証する。

DR-net や RAID システムについて、規模が小さい場合の具体的な MTTF を比較することにより、それぞれの信頼性を示す。また、一般的な規模についてそれぞれのシステムのマスク可能なディスク故障数について議論する。さらに信頼性と記憶効率の面からシステムの柔軟な構成について考察する。システムが保持する冗長情報の割合を低減したいくつかの構成例を示し、それらの信頼性についても評価する。

4.1 ディスク故障とデータの喪失

ディスク故障により DR-net のデータが失われる場合は大きく 2 つに分類できる。一つは、図 4.1のように再構築に必要なパリティディスクが、FPG を使ったときも SPG を使ったときも故障している場合である。図 4.1の例ではノード 22 の故障ディスクのデータを再構築するとき、FPG を使っても SPG を使ってもそれぞれ必要なパリティノード 32, 11 のディスクが故障しているため、再構築できない。

もう一つは、図 4.2のように、再構築に自分自身のデータが必要となるときである。図 4.2の例では、ノード 02 のデータを SPG を用いて再構築しようとすると、ノード 22, 33 の再構築のために FPG, SPG を利用し、最終的にノード 13 の再構築ではノード 02 自身のデータが必要となるため、再構築できない。

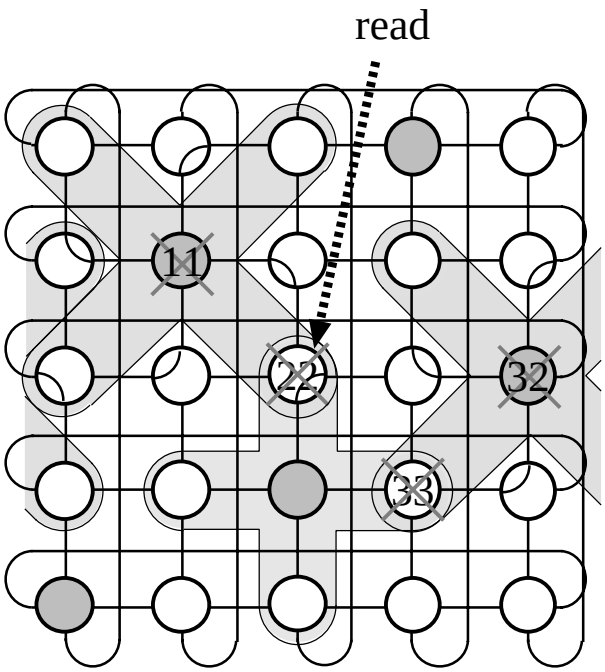

図 4.1: 再構築不可能な場合 1

4.2 MTTF による評価

本節では、25 台のディスクノードを持つ DR-net システムの MTTF を同程度の規模の各 RAID システムと比較し、DR-net が高い信頼性を有することを示す。RAID レベル 2 については、記憶効率が悪く、また対応できるディスク故障数がレベル 3 から 5 と同じであるから、比較の対象からは除外する。

4.2.1 信頼性モデル

信頼性モデルとしては、[26] において各パリティグループ毎にディスク故障が発生したときのデータ再構築が不可能となる確率をもとにシステム全体の信頼性を近似式で求める方法があげられている。しかし、DR-net では 2 種類のパリティグループがあるため、1 つのパリティグループで再構築不可能であっても他方のパリティグループで再構築可能となる場合があり、この方法は適さない。

このように、パリティグループ毎に信頼性を確定できないような構成に対して、[10] では

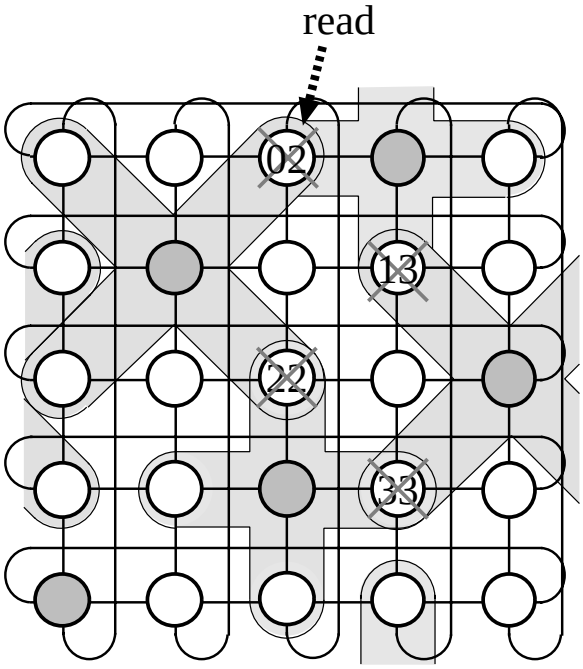

図 4.2: 再構築不可能な場合 2

モンテカルロシミュレーションによる方法が用いられている。一方、[3][36] ではマルコフ過程に基づく方法が採用されている。この方法ではパリティグループ毎の再構築不可能な確率を求める必要はなく、ディスク故障数に関してシステム全体の再構築可能な確率 (coverage) を求めればよい。ディスク故障数に関するカバリッジを任意のシステム規模について解析的に求めることは困難であるが、全ての故障パタンの中からデータ再構築可能なパタン数を計算機で数えあげることによりカバリッジを算出することができる。ここでは、この方法に基づく信頼性比較を行なう。

全体のディスク数を N, 各ディスクの故障率を λ, 各ディスクの修理率を μ, i 個目のディスク故障が起きたときに全ての故障ディスクのデータが再構築可能な状態にある確率 (カバリッジ) を c_i とすると、システムのモデルは図 4.3のようになる。図中の S_i はシステム内に i 個のディスク故障が発生し、なおかつ全ての故障ディスクのデータが再構築可能である状態, S_F はデータ再構築が不可能なディスク故障が発生した状態である。n は最大 n 個のディスク故障でもデータは失われないことを表す。

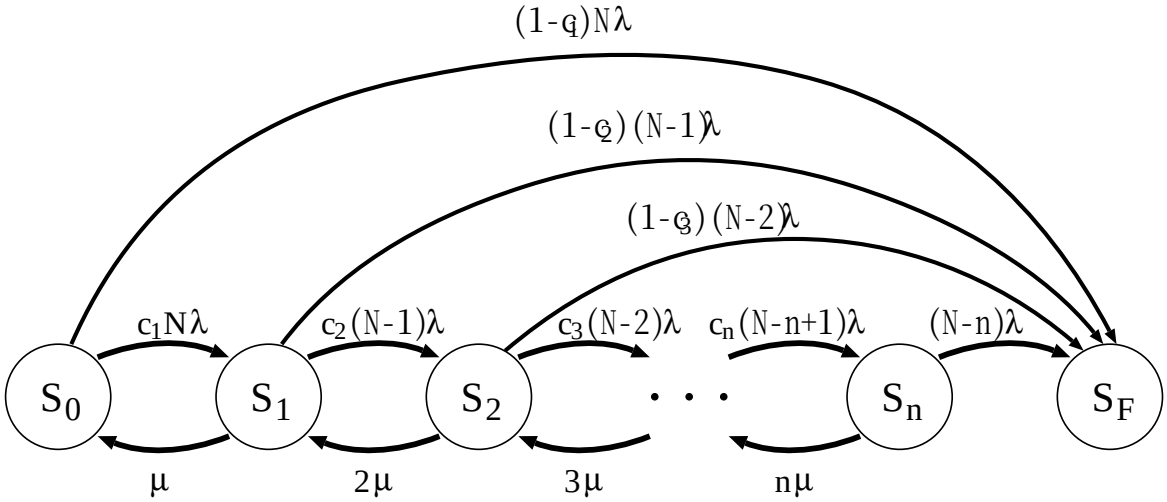

図 4.3: マルコフモデル

システムが時刻 t において状態 S_i である確率を $p_i(t)$ とすると、次の関係が成り立つ。

$$\begin{aligned}
\frac{dp_0(t)}{dt} &= -N\lambda p_0(t) + \mu p_1(t) \\
\frac{dp_i(t)}{dt} &= c_i(N-i+1)\lambda p_{i-1}(t) - \{(N-i)\lambda + i\mu\}p_i(t) + (i+1)\mu p_{i+1}(t) \qquad (0 < i < n) \\
\frac{dp_n(t)}{dt} &= c_n(N-n+1)\lambda p_{n-1}(t) - \{(N-n)\lambda + n\mu\}p_n(t)
\end{aligned}$$

[3] では c_i を 0 または 1 として $\mu \neq 0$ の場合の近似式を扱っているが、DR-net では c_i は i に応じて 0 から 1 までさまざまな値をとるため、[3] の手法をそのまま使えない。ここでは、簡単のために $\mu = 0$、つまり故障ディスクが修理されないときの信頼性を考える。$\mu = 0$ とすると、

$$\begin{aligned}
\frac{dp_0(t)}{dt} &= -N\lambda p_0(t) \\
\frac{dp_i(t)}{dt} &= c_i(N-i+1)\lambda p_{i-1}(t) - (N-i)\lambda p_i(t) \qquad (0 < i \leq n)
\end{aligned}$$

また、システムの信頼度 $R(t)$ は

$$R(t) = \sum_{i=0}^{n} p_i(t)$$

であるから、λ を一定とするとシステムの MTTF は

$$\begin{aligned} MTTF &= \int_0^\infty R(t)\,dt \\ &= \sum_{i=0}^{n} \int_0^\infty p_i(t)\,dt \\ &= \frac{1}{N\lambda} + \sum_{i=1}^{n} \left(\prod_{j=1}^{i} c_j \right) \frac{1}{(N-i)\lambda} \end{aligned} \tag{4.1}$$

となる。

4.2.2 各構成の MTTF

前節の信頼性モデルに基づき、各システムの MTTF を示す。

RAID レベル 0

レベル 0 ではいかなるディスク故障でもデータが失われるので、$n = 0$ である。従って、

$$MTTF_{\text{RAID0}} = \frac{1}{N\lambda}$$

となる。

RAID レベル 1

レベル 1 では 2 重化したディスクが両方とも故障するとデータが失われる。全体のディスク数を $N = 2m$ とすると、最大で m 個の故障に耐えられるから $n = m$ である。また、各 c_i は i 個の故障が存在するときの再構築可能なパタン数を i 個の故障が存在するときの全てのパタン数で割ればよい。i 個のディスクが故障しているとき、故障ディスクのデータが再構築可能であるためには各故障ディスクは異なる 2 重化のペアに存在しなければならない。全体で 2 重化のペアは m 個あり、2 つのディスクのうちどちらが故障していてもよいから、i 個の故障が存在するときのデータが失われないパタン数は ${}_mC_i \cdot 2^i$ である。従って、c_i は

$$\frac{{}_mC_i \cdot 2^i}{{}_NC_i}$$

となる。これを式 (4.1) に代入すると

$$\begin{aligned} MTTF_{\text{RAID1}} &= \frac{1}{N\lambda} + \sum_{i=1}^{m} \left(\prod_{j=1}^{i} c_j \right) \frac{1}{(N-i)\lambda} \\ &= \frac{1}{N\lambda} + \sum_{i=1}^{m} \left(\prod_{j=1}^{i} \frac{{}_mC_j \cdot 2^j}{{}_NC_j} \right) \frac{1}{(N-i)\lambda} \end{aligned}$$

となる。

RAID レベル 3 〜 5

レベル 3 〜 5 はいずれもパリティグループ内でのいかなる単一ディスク故障にも耐えられるが、1 つのパリティグループ内での任意の 2 つのディスク故障でデータが失われる。

ディスク故障時に、データ再構築の負荷を分散させるような Declustered Parity[19][22] 構成では、任意の 2 つのディスク故障でデータが失われる。この場合、$N_G = 1, G = N$ と考えられるから、$n = 1, c_1 = 1$ とすると

$$\begin{aligned} MTTF_{\text{decRAID5}} &= \frac{1}{N\lambda} + \frac{1}{(N-1)\lambda} \\ &= \frac{2N-1}{N(N-1)\lambda} \end{aligned}$$

となる。

Declustered Parity でないときを考えると、パリティグループサイズを G, パリティグループ数を N_G としたとき、$n = N_G, N = G \times N_G$ である。また c_i は

$$c_i = \frac{{}_{N_G}C_i G^i}{{}_NC_i}$$

となる。従って MTTF は

$$MTTF_{\text{nondecRAID5}} = \frac{1}{N\lambda} + \sum_{i=1}^{N_G} \left(\prod_{j=1}^{i} \frac{{}_{N_G}C_i G^i}{{}_NC_i} \right) \frac{1}{(N-i)\lambda}$$

となる。

RAID レベル 6, EVENODD

RAID レベル 6 や EVENODD[1][2] は、パリティグループ内での任意の 2 つのディスク故障に耐えられる。

表 4.1: Declustered Parity でないときの RAID6, EVENODD のカバリッジ ($N_G = 4, G = 6$)

故障数 (i)	全故障パタン数	再構築可能パタン数	$\prod_{j=1}^{i} c_i$
1	24	24	1.00
2	276	276	1.00
3	2024	1944	0.960
4	10626	9126	0.859
5	42504	29160	0.686
6	134596	62100	0.461
7	346104	81000	0.234
8	735471	50625	0.0689
9	1225785	0	0.00

Declustered Parity 構成では任意の 3 つのディスク故障でデータが失われるから、$n = 2$, $c_1 = c_2 = 1$ であるから、

$$\begin{aligned} MTTF_{\text{decRAID6}} &= \frac{1}{N\lambda} + \frac{1}{(N-1)\lambda} + \frac{1}{(N-2)\lambda} \\ &= \frac{3N^2 - 6N + 2}{N(N-1)(N-2)\lambda} \end{aligned}$$

となる。

Declustered Parity でないときは、パリティグループサイズを G, パリティグループ数を N_G とすると、$n = 2N_G$ である。しかし、一般の G, N_G について c_i は容易には求まらないため、ここでは $N_G = 4, G = 6$ の場合 ($N = G \times N_G = 24$) に各 i についてデータが失われないパタンを数え挙げることにより c_i を求めた。式 (4.1) より MTTF は

$$\begin{aligned} MTTF_{\text{nondecRAID6}} &= \frac{1}{24\lambda} + \sum_{i=1}^{8} \left(\prod_{j=1}^{i} c_j \right) \frac{1}{(24-i)\lambda} \\ &= \frac{0.299}{\lambda} \end{aligned}$$

である。

表 4.2: 5×5 構成の DR-net のカバリッジ (固定パリティノード)

故障数 (i)	全パタン数	再構築可能パタン数	$\prod_{j=1}^{i} c_j$
1	25	25	1.00
2	300	300	1.00
3	2300	2280	0.991
4	12650	12070	0.954
5	53130	45870	0.863
6	177100	124400	0.702
7	480700	232500	0.484
8	1081575	274545	0.254
9	2042975	158625	0.0776
10	3268760	0	0.00

DR-net

DR-net の場合にも一般の N について c_i を求めるのは困難である。ここでは、5×5 の構成について考える。

パリティを分散させない場合と、MPG, MPN によってそれぞれ分散させた場合のカバリッジは表 4.2, 4.3, 4.4となり、n はそれぞれ $n = 9$, $n = 7$, $n = 8$ となる。

それぞれの MTTF を算出すると

$$
\begin{aligned}
MTTF_{\text{DR-net}} &= \frac{0.34}{\lambda} \\
MTTF_{\text{DR-net(MPG)}} &= \frac{0.24}{\lambda} \\
MTTF_{\text{DR-net(MPN)}} &= \frac{0.2\ 5}{\lambda}
\end{aligned}
$$

となる。

4.2.3 MTTF の比較

以上の各方式の $N = 2\ 4\ 2\ 5$における MTTF を長い順にまとめると、表 4.5 になる。表中の D.P. は Declustered Parity 構成を表す。また、冗長率が括弧で括ってあるものは他と異な

表 4.3: 5×5 構成の DR-net のカバリッジ (MPG)

故障数 (i)	全パタン数	再構築可能パタン数	$\prod_{j=1}^{i} c_j$
1	25	25	1.00
2	300	300	1.00
3	2300	2200	0.957
4	12650	9950	0.787
5	53130	24080	0.453
6	177100	23050	0.130
7	480700	5050	0.0105
8	1081575	0	0.00

表 4.4: 5×5 構成の DR-net のカバリッジ (MPN)

故障数 (i)	全パタン数	再構築可能パタン数	$\prod_{j=1}^{i} c_j$
1	25	25	1.00
2	300	300	1.00
3	2300	2200	0.957
4	12650	10250	0.810
5	53130	28880	0.544
6	177100	44400	0.251
7	480700	31100	0.0647
8	1081575	5525	0.00511
9	2042975	0	0.00

表 4.5: 各方式の MTTF

構成	ディスク数	冗長率	MTTF
DR-net	25	1/3	$0.34/\lambda$
RAID レベル 1	24	(1/2)	$0.300/\lambda$
RAID レベル 6	24	1/3	$0.299/\lambda$
DR-net(MPN)	25	1/3	$0.25/\lambda$
DR-net(MPG)	25	1/3	$0.24/\lambda$
RAID3 〜 5	24	1/3	$0.22/\lambda$
RAID6(D.P.)	25	1/3	$0.13/\lambda$
RAID3 〜 5(D.P.)	25	1/3	$0.082/\lambda$
RAID0	25	(0)	$0.040/\lambda$

る冗長率である。RAID レベル 1 では N は偶数でなければならないので $N = 24$ とした。また、DR-net と冗長率をそろえるために、Declustered Parity 構成ではないときの RAID レベル 3 〜 5 では $N_G = 8, G = 3, N = N_G \times G = 2\ 4$とした。同様に、Declustered Parity 構成ではないときの RAID レベル 6 では $N_G = 4, G = 6$,$N = N_G \times G = 2\ 4$とした。値はもっとも高い信頼性を示すように (G が最小となるように) 選んだ。

この表から DR-net でパリティ分散を行なわない場合、どの方式よりも高い信頼性を有することがわかる。また、MPG, MPN によるパリティ分散を行なった場合でも、Declustered Parity 構成ではないときの RAID レベル 3 〜 5 よりも高い信頼性が得られる。

パリティを分散して配置している場合には、フェーズ毎に、MPG ではパリティグループが、MPN ではパリティノードが移動するため、1 つの物理的な故障パタンがアクセスのフェーズ毎に論理的に異なるパタンとなる。それらのうち、1 つでもデータの失われるパタンに当てはまるフェーズがあれば、データは失われる。従って、パリティを分散した場合には、固定して配置した場合よりも信頼性が低くなる。また、MPG と MPN の比較では、MPN の方が若干良い結果が出ている。MPN ではフェーズ毎にパリティグループ内でパリティノードの位置が変化することにより、より多くの図 4.1のようなパタンに当てはまる可能性がある。MPG ではそれに加えて、フェーズ毎にパリティグループが移動するため 1 つのデータディスクの再構築に必要なパリティグループがフェーズ毎に変化し、図 4.2のようなパタンに当てはまる可能性も増加する。このことにより、MPG よりも MPN によるパリティ分散の方が高い信頼性を得ることができる。

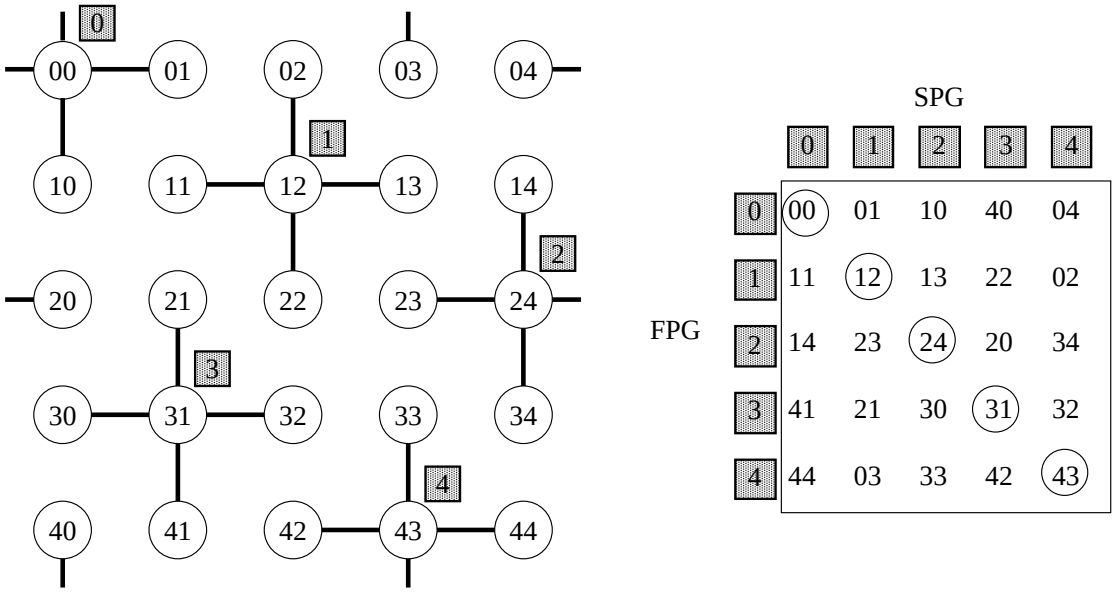

図 4.4: システム構成の行列表現

4.3 一般的な規模の信頼性に関する考察

前節では、25 台という具体的なディスクノード数について MTTF を評価したが、本節では一般的な規模の信頼性について議論する。

4.3.1 マスク可能なディスク故障の数

式 (4.1) から、MTTF を計算するためにはマスク可能な最大ディスク故障数とそれぞれの故障数におけるカバリッジを知る必要がある。大規模な構成の DR-net については、カバリッジを求めることが困難であるため、MTTF の計算はできない。そのため、マスク可能なディスク故障数 (以後、「マスク数」と呼ぶ) についてのみ議論することにより、RAID と比較したときの DR-net の一般的な信頼性に関して考察する。マスク数についても、任意の構成の DR-net について一般的に求めるのは困難であるが、ここでは十字型のパリティグループを用いる $5 \times n$ のトーラス構成について述べる。

システム構成の行列表現

DR-net で、ある故障パタンでデータが失われるかどうかを考えるとき、実際のネット

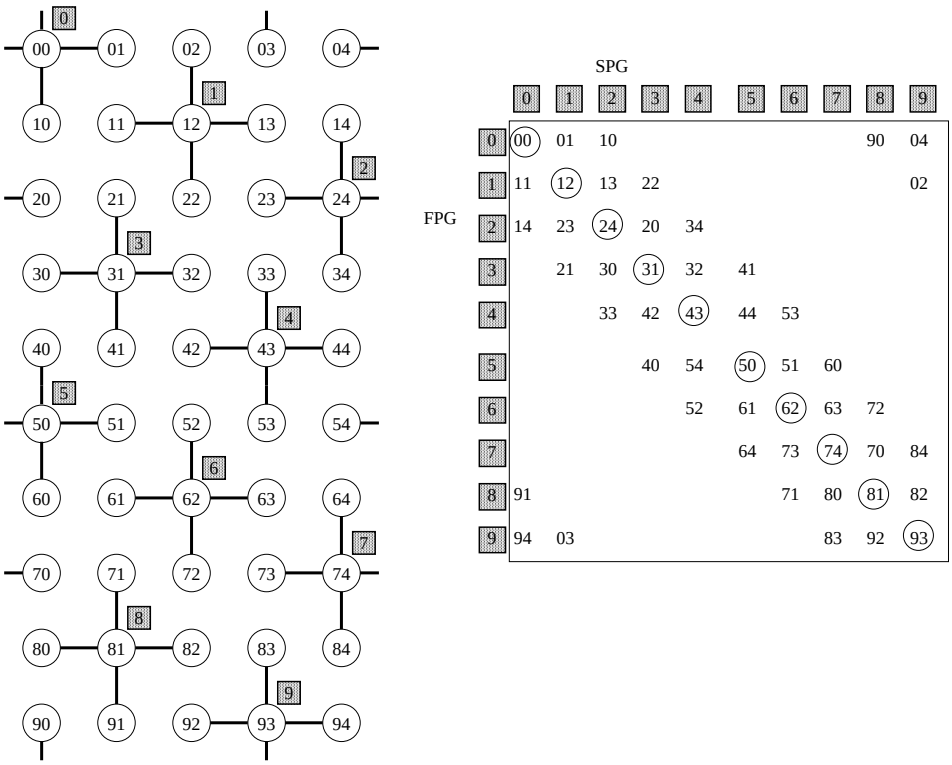

図 4.5: 5×10 の構成の行列表現

ワークの結合形態で考えると、FPG と SPG が互いに重なりあい、また FPG 同士あるいは SPG 同士の関係も直観的に分かりにくい。そこで、パリティグループ間の関係を整理するために、システムをネットワークトポロジから切り離して表現することを考える。

FPG, SPG にそれぞれ番号を付け、i 番めの FPG, SPG をそれぞれ F_i, S_i で表す。このとき、FPG を行、SPG を列として、それぞれのパリティグループに属すノードを行列の要素として配置する。簡単のため、パリティノードは中心ノードに固定されているとすると、行列の対角線上の要素がパリティノードに対応する。図 4.4は 5×5 の構成を行列表現した例である。ノード数が N の場合には、$N/5 \times N/5$ の正方行列となる。

ここで、$5 \times n$ のトーラス構成を考えると、パリティグループ数 g は $g = 5n/5 = n$ となる。便宜上、ネットワークの横の長さが 5 であるとする。図 4.4と同様にパリティグループを配置し、パリティグループ番号をネットワークの上から順番につけると、各 FPG, SPG の間に次のような性質が成り立つ。F_i に属すノードが、

- 十字型の上方に位置するとき、そのノードは S_{i-2} に属す。

- 十字型の下方に位置するとき、そのノードは S_{i+2} に属す。
- 十字型の右側に位置するとき、そのノードは S_{i+1} に属す。
- 十字型の左側に位置するとき、そのノードは S_{i-1} に属す。
- 十字型の中心に位置するとき、そのノードは S_i に属す。

ただし、加減算は g を法とする。ネットワークの横幅が 5 である限り、ネットワーク規模は隣接するパリティグループの位置関係に影響しないため、常にこの性質が成り立つ。従って、行列表現では対角線を中心とした部分に要素が集まる。図 4.5は 5×10 のトーラス構成を行列表現した例である。

マスク可能な故障パタン

ある故障パタンの全ての故障がマスクできるかどうかは、次の規則を用いて確かめることができる。これらの規則は 4.1節の 2 つの場合にそれぞれ対応する。

規則 1 同一行あるいは同一列に存在する故障ノードを結んだときに、ループができれば故障はマスクできない。

規則 2 同一行あるいは同一列に存在する故障ノードを結んだときに、両端がパリティノードであれば、故障はマスクできない。

図 4.6にマスクできないパタンの例を示す。この表現を用いて、マスク数を求めるには、上記の規則からマスクできないパタンを避け、かつ故障数が最大になるようなパタンを見つければよい。$5 \times n$ のトーラス構成では、前述のように対角線を中心とした要素を持つ行列表現となるため、上記の規則に当てはまらないパタンとして、図 4.7のようなものが考えられる。この場合のマスク数は

$$2g - 1$$

となる。$2g - 1$ が最大のマスク数であること示すためには証明が必要であるが、証明がなくとも実際に存在するパタンであることから、$2g - 1$ は $5 \times n$ のトーラスネットワークを用いる構成での最大マスク数の下限であり、$5 \times n$ の構成での最大マスク数は、少なくとも $2g - 1$ 以上である。$g = 5$、すなわち 5×5 のトーラス構成では $2g - 1 = 9$ となり、前節で示した最大マスク数に一致する。

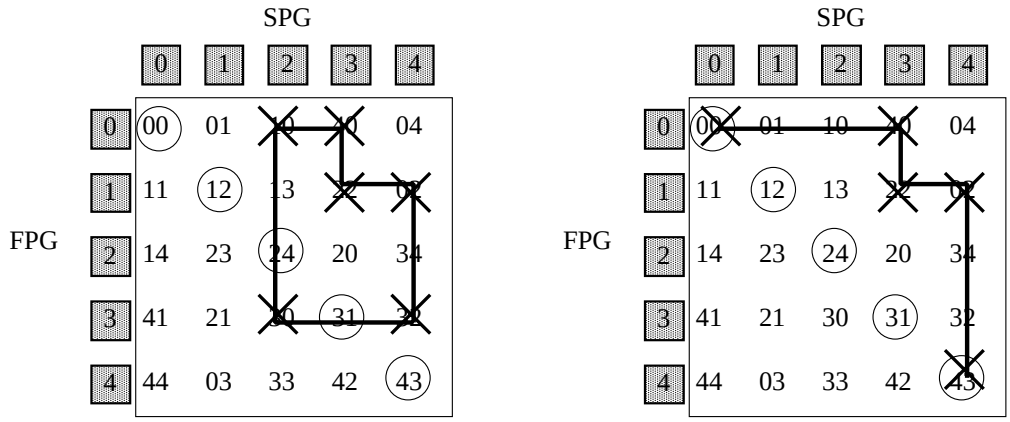

図 4.6: マスクできない故障パタンの例

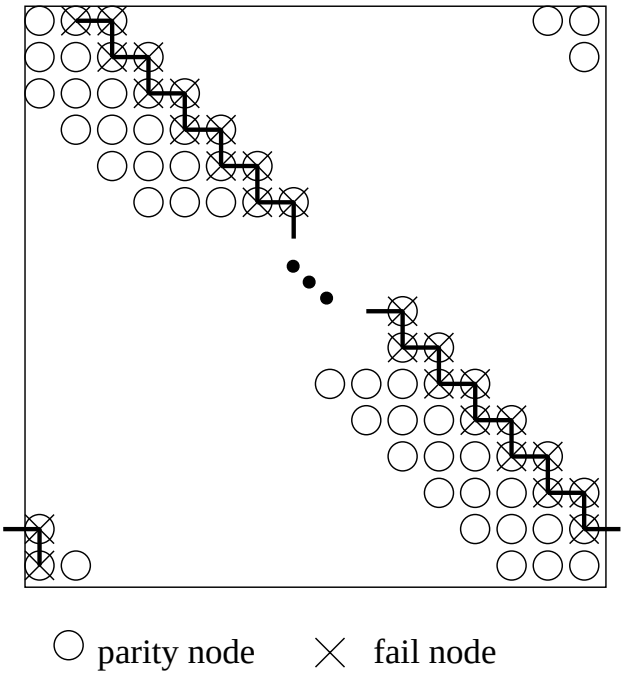

図 4.7: マスク可能な故障パタン (パリティ固定)

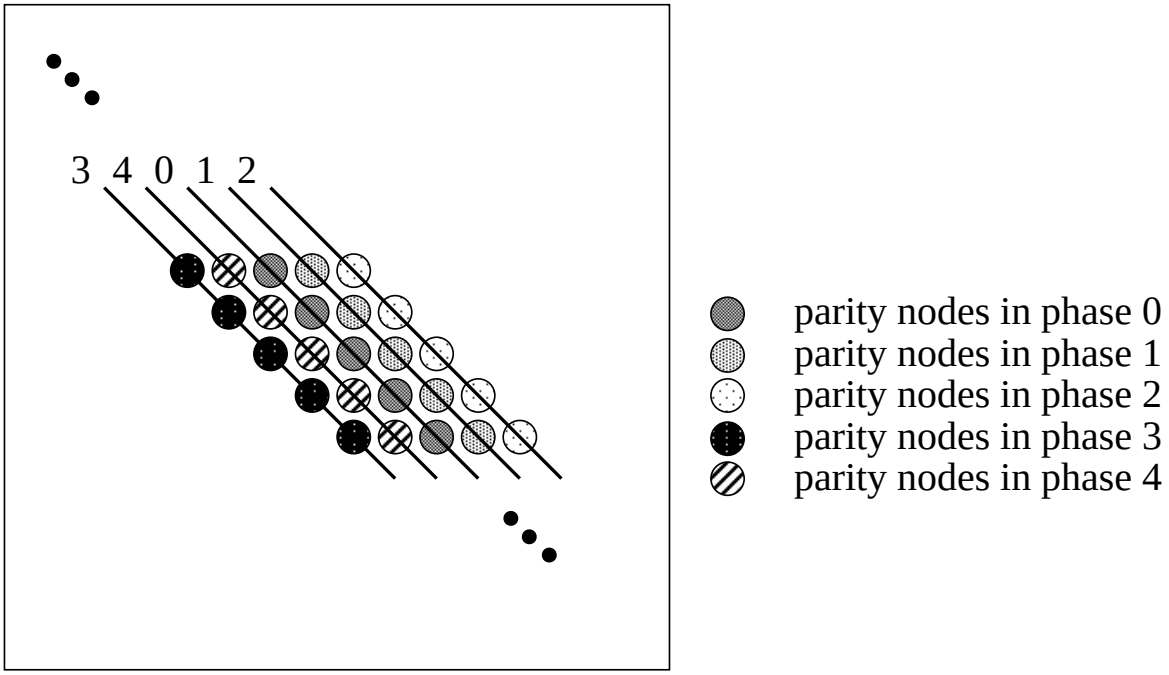

図 4.8: MPN での各フェーズでのパリティノード

MPN によるパリティ分散

MPN でパリティを分散保持した場合には、各ノードがそれぞれパリティノードとなるフェーズがある。そのため、固定パリティではマスク可能な故障パタンでも、前述の規則 2 に該当する場合がある。各ノードがどのようなフェーズでパリティノードになるかを調べると、図 4.8のようになる。

ここで、全てのフェーズで前述の規則に抵触せず、なるべく故障数の多い故障パタンとして図 4.9のようなものが考えられる。このパタンでのマスク数は、6 行毎に 10 個のディスク故障が存在し、最後の l 行 ($l < 6$) には $2(l-1)$ 個のディスク故障が存在するから、x を $g/6$ の整数部として、

$$x \times 10 + (g - 6x - 1) \times 2 = 2g - 2x - 2$$

となる。この値は、MPN を用いた場合の最大マスク数の下限である。$g = 5$ の場合には $2g - 2x - 2 = 8$ で、最大マスク数に一致する。

なお、MPG ではフェーズ毎にパリティグループの構成ノード自体が変化するため、行列表現に基づく手法ではマスク数を求めることはできない。

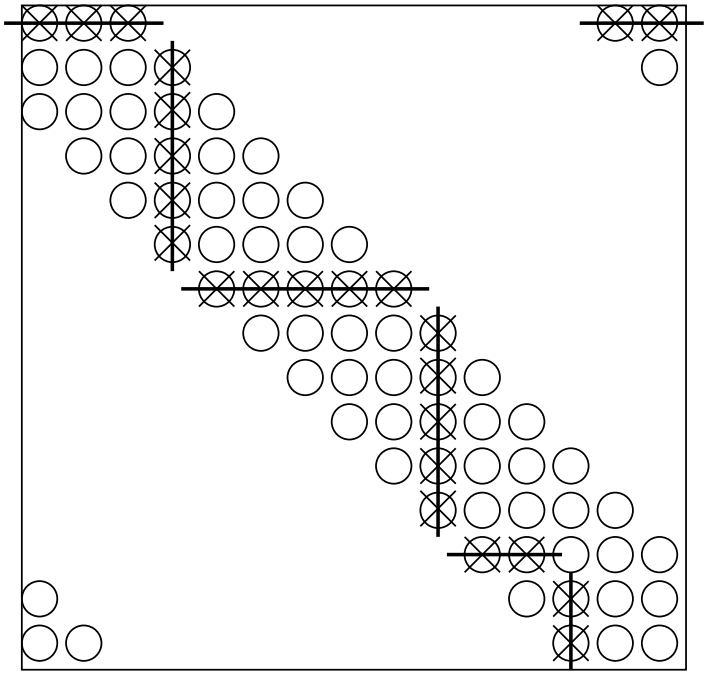

図 4.9: マスク可能な故障パタン (MPN)

4.3.2 RAID との比較

DR-net と冗長率が同等の場合、レベル 3 〜 5 では $g = 3$、レベル 6 では $g = 6$ となるから、各レベルの RAID の最大マスク数 m は以下のようになる。

レベル 1 $N/2$

レベル 3 〜 5 $g = N/3$

レベル 6 $2g = 2N/6 = N/3$

DR-net では $g = 5$ であるから、最大マスク数の下限は、パリティを固定した場合では

$$m = 2g - 1 = \frac{2N}{5} - 1$$

MPN で分散させた場合には

$$\begin{aligned} m &= 2g - 2x - 2 \\ &= \tfrac{2N}{5} - 2x - 2 \end{aligned}$$

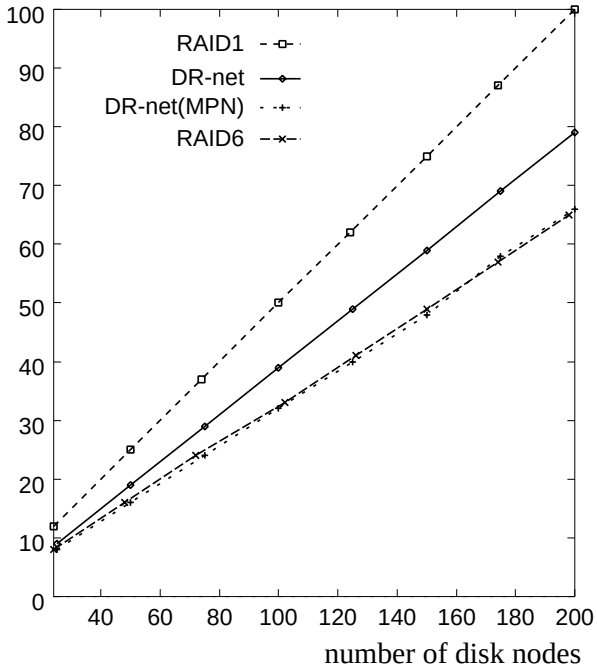

図 4.10: 各構成規模での最大マスク数の比較

となる。これらの比較を図 4.10に示す。ただし、図中の DR-net の値は最大マスク数の下限である。この結果、構成規模に依存せず、パリティを固定した DR-net では RAID レベル 6 よりも多くの故障をマスクすることが可能であり、MPN でパリティを分散した場合でもほぼ同等の数の故障をマスクできることが分かる。

4.4 記憶効率の改善と信頼性

一般に、冗長情報を増やすことによりシステムの信頼性を向上させることができるが、冗長率の増加は記憶効率の悪化を招く。レベル 1 以外の RAID システムは、パリティグループ内のデータディスクの数を容易に変更することができ、信頼性と記憶効率のトレードオフを自由に調整できる。一方、これまでに議論してきた構成の DR-net は、データと冗長情報の比率は 2:1 で固定されている。DR-net においてもユーザの要求に合わせて信頼性と記憶効率を柔軟に選択できることが望ましい。その際、冗長率を変更した構成でも 2 つのディスク故障でデータが失われないことが重要である。

本節では、DR-net の冗長率の変更についてその指針を示し、記憶効率を改善したいくつ

かの具体的な構成についてその信頼性を評価する。

4.4.1 冗長率の変更

冗長率を変更するためには、1 つのパリティグループ内に含まれるディスクノード数を変更すればよいが、その際に、任意の 2 つのディスク故障に対してデータの喪失を防ぎ、パリティグループ内のノード間通信のコストを抑えることが望ましい。従って、DR-net で用いるパリティグループを設計する際には、次の 2 つのことについて注意する必要がある。

信頼性: 1 パリティグループ内のノード数および 1 ノードが属すパリティグループ数

性能: パリティグループ内での通信の局所性を考慮した規則的なパリティグループの形状

1 つのノードが必ず 2 つのパリティグループに属すこと、および各パリティグループ内のノード数, 全体のノード数を簡潔に表現するために、前節で述べた行列表現を利用することができる。例えば、1 パリティグループが 7 ノードで全体のノード数が 63 の構成は、図 4.11のような行列から得られる。ネットワーク上でのパリティグループの配置は、行列内の各ノードに適切なノード位置を割り当てることにより実現できる。ただし、パリティグループ内での通信の局所性を考慮し、さまざまなネットワークトポロジに対して一般的な割り当て方法を示すことは困難であるため、必ずしも適切な配置が見つけられるとは限らない。図 4.11の構成は 2 次元トーラスネットワーク上では図 4.12のように実現できる。

行列表現を用いることにより冗長率の変更は柔軟に行なえるが、パリティグループ内での通信の局所性を確保することが課題となる。従って、通信の局所性を重視する場合には、まずネットワーク上でのパリティグループの配置を優先して、なるべく要求に近い冗長率を持つ構成を ad hoc に探すようなアプローチも考えられる。

以下では、実際に構成を挙げることにより、通信の局所性を保ったまま記憶効率の改善が可能であることを示し、またその信頼性について検証する。

4.4.2 構成例

ここで示す 3 つのパリティグループ構成は、いずれも 2 次元トーラスネットワーク上でグループ内での通信の局所性を保ちつつ、従来の構成に比べて冗長率を低減し記憶効率を改善している。

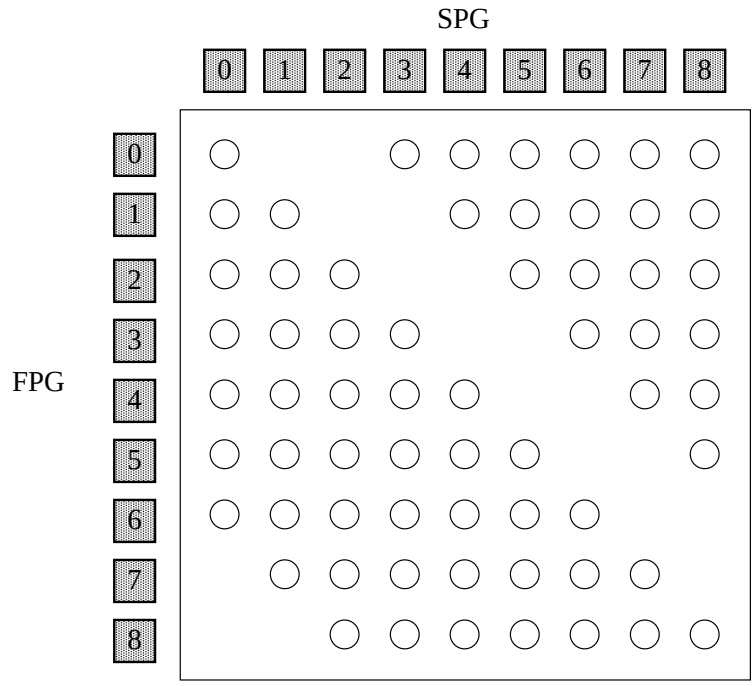

図 4.11: ノード数 63 の構成の行列表現 (7 ノード/パリティグループ)

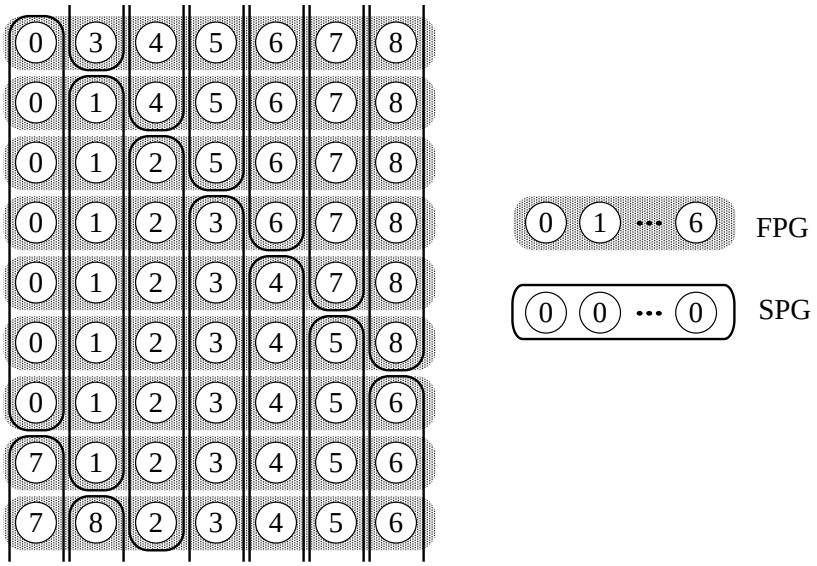

図 4.12: ノード数 63 のトーラス構成 (7 ノード/パリティグループ)

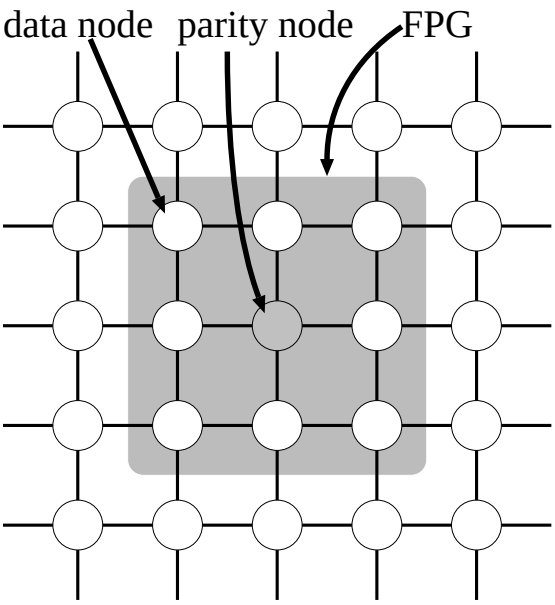

図 4.13: 9-neighbors の FPG

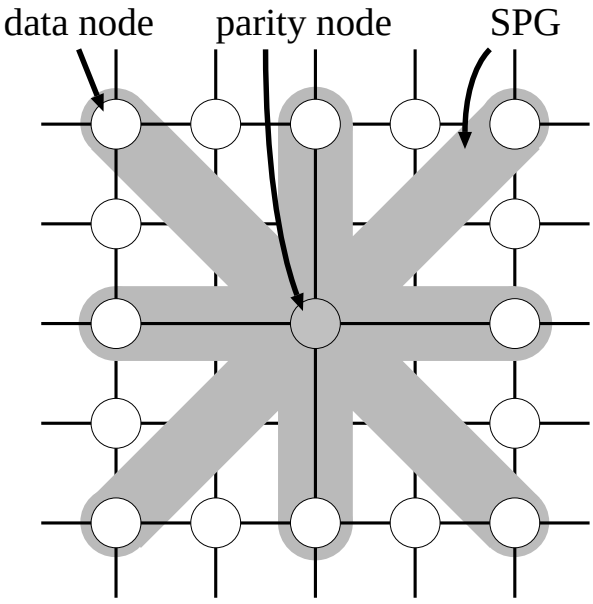

図 4.14: 9-neighbors の SPG

9-neighbors

図 4.13, 4.14に 9-neighbors と呼ばれるパリティグループ構成の FPG, SPG を示す。FPG, SPG ともに 9 つのノードから構成される。この形状を用いた場合トーラスネットワークの一辺のノード数は 3 の倍数でなければならない。各データノードからパリティノードまでの平均距離は FPG で 1.5, SPG で 3 である (この値は、ネットワークに斜め方向のリンクを追加することによって、それぞれ 1, 2 となる)。9-neighbors での冗長率は、データディスク 8 台に対し FPG と SPG のパリティディスクがそれぞれ 1 台ずつあるから、$2/(8+2) = 1/5$ である。

9-neighbors では従来と同様に MPG,MPN によるパリティ分散が可能である (図 4.15, 4.16)。パリティ分散のフェーズ数は 9 で、各ノードはいずれかのフェーズで 1 回だけパリティノードになる。MPN を用いた場合のデータノードからパリティノードまでの平均距離は FPG で 2, SPG で 4 である。

9-neighbors を用いた DR-net の MTTF を 6×6 の構成について求めると

$$
\begin{aligned}
MTTF_{\text{9N}} &= \frac{0.16}{\lambda} \\
MTTF_{\text{9N(MPG)}} &= \frac{0.095}{\lambda} \\
MTTF_{\text{9N(MPN)}} &= \frac{0.14}{\lambda}
\end{aligned}
$$

となる。表 4.6に同程度の冗長率を持つ他の方式との比較を示す。declustered parity でないときの RAID レベル 3 〜 5 は $N_G = 7, G = 5$、また RAID レベル 6 は $N_G = 3, G = 1\,0$とした。表 4.6で、9-neighbors を用いた DR-net は RAID レベル 6 よりもかなり低い信頼性となっているが、原因のひとつは 6×6 の構成では 2 つの故障によってデータが失われる場合があるためである (図 4.17)。この問題はトーラスネットワークの一辺が 9 以上であれば解消し、いかなる 2 つの故障でもデータは失われなくなる。また、比較した RAID レベル 6 はディスク数が 30 であるのに対し、DR-net は 36 であることを考えると、表に示されるほどの差はないと考えられる。

9-neighbors の構成法は、3 次元以上のトーラスにも適用できる。その場合、1 つ 1 つのパリティグループが大きくなり信頼性が低下するが、よりよい記憶効率を得ることができる。

overlapped 9-neighbors

前述の 9-neighbors および従来のパリティグループを用いた DR-net は、FPG, SPG という 2 種類の異なるグループを重ね合わせ、両者がパリティノードを共有している。2 つの

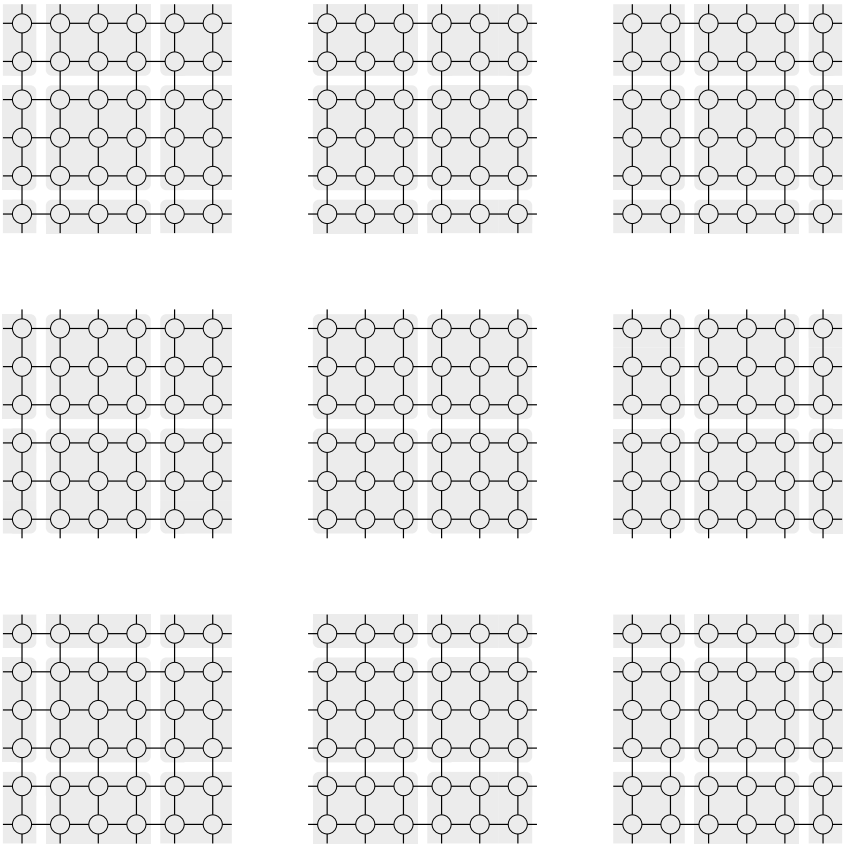

図 4.15: 9-neighbors を用いた MPG フェーズ

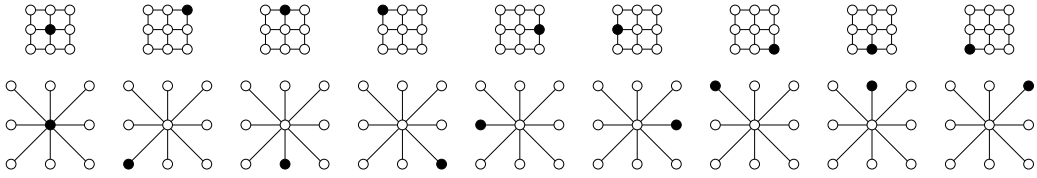

図 4.16: 9-neighbors を用いた MPN フェーズ

表 4.6: 9-neighbors を用いた DR-net と各方式の MTTF の比較

構成	ディスク数	冗長率	MTTF
RAID レベル 1	36	(1/2)	$0.24/\lambda$
RAID レベル 6, EVENODD	30	1/5	$0.19/\lambda$
DR-net(9N)	36	1/5	$0.16/\lambda$
DR-net(9N-MPN)	36	1/5	$0.14/\lambda$
RAID レベル 3 〜 5	35	1/5	$0.13/\lambda$
DR-net(9N-MPG)	36	1/5	$0.095/\lambda$
RAID レベル 6, EVENODD(D.P.)	36	1/5	$0.086/\lambda$
RAID レベル 3 〜 5(D.P.)	36	1/5	$0.056/\lambda$
RAID レベル 0	36	(0)	$0.028/\lambda$

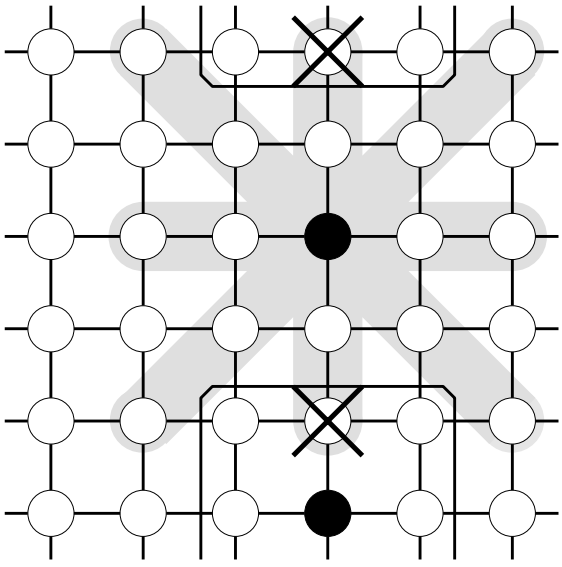

図 4.17: 9-neighbors を用いた 6×6 構成の DR-net において 2 つの故障でデータが失われる例

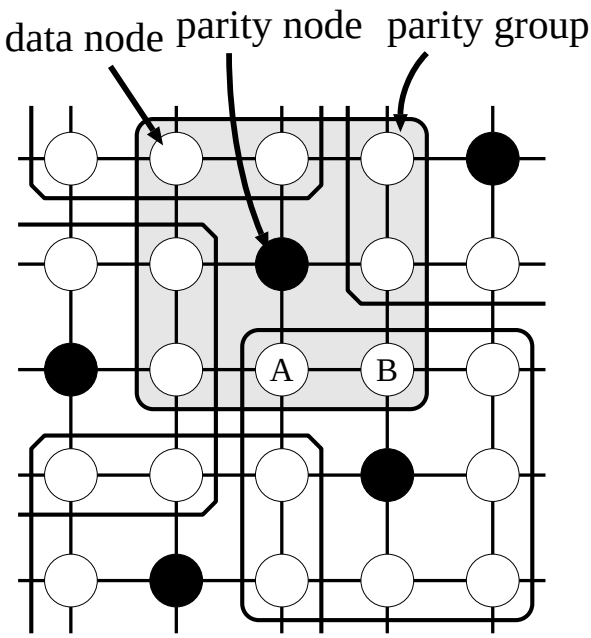

図 4.18: overlapped 9-neighbors

パリティグループがデータノードを共有するのは信頼性を向上を目的として冗長率をあげるためであるが、パリティノードを共有するとパリティディスクは FPG と SPG のパリティを両方保持しなければならず、データディスクに比べて 2 倍の容量が必要になる。さらに、パリティディスクが故障すると FPG, SPG の 2 つのパリティグループの冗長情報が同時に失われるため、パリティノードの共有は信頼性に関して不利である。

そこで、FPG, SPG という 2 種類のパリティグループを重ね合わせるのではなく、互いに異なるパリティノードを持つ 1 種類のパリティグループを重ね合わせ、データノードがそれぞれ 2 つのパリティグループに属すようにすることが考えられる。このようにすると、パリティノードは 1 つのパリティグループに属すがデータノードは 2 つのパリティグループに属すことになる。この方法ではパリティを MPN によって分散保持することはできず MPG を使うことになる。

図 4.18に overlapped 9-neighbors を示す。この形状を用いた場合トーラスネットワークの一辺のノード数は 5 の倍数でなければならない。各データノードからパリティノードまでの平均距離は 1.5 である。overlapped 9-neighbors の冗長率は $1/5$ である。

overlapped 9-neighbors で MPN を用いたパリティ分散しようとすると、データノードが 2 つのパリティグループに属すため、2 つのパリティグループ間でパリティの保持領域が衝突する。従って MPN は適用できず、パリティ分散は MPG によって行なう。

overlapped 9-neighbors は 9-neighbors と同じ冗長率であるが、9-neighbors に比べてデータ/パリティノード間の平均距離を小さくし、グループ内の通信のローカリティを高めている。反面、2 つのパリティグループが 2 つのデータノードを共有しているため、その 2 つのノードのディスクが故障すると再構築が不可能となり、データが失われる。図 4.18の例ではノード A とノード B のディスクが共に故障したような場合にデータが失われる。

overlapped 9-neighbors を用いた DR-net の MTTF を 5×5 の構成について求めると、

$$
\begin{aligned}
MTTF_{\text{O9N}} &= \frac{0.21}{\lambda} \\
MTTF_{\text{O9N(MPG)}} &= \frac{0.15}{\lambda}
\end{aligned}
$$

となる。表 4.7に同程度の冗長率を持つ他の方式との比較を示す。declustered parity でないときの RAID レベル 3 〜 5 は $N_G = 9, G = 4$、また RAID レベル 6 は $N_G = 3, G = 1\,0$とした。

2 つのパリティグループが 1 つではなく 2 つのデータノードを共有することで、信頼性がかなり落ちていることがわかる。

overlapped 7-neighbors

あるノードをパリティノード, またそれに隣接する 8 つのノードをデータノードとして、その 3×3 の正方形領域でパリティグループを形成し、それらの辺と辺をそれぞれ重ね合わせると、正方形の角に位置するデータノードは 4 つのパリティグループに属すことになる。角のデータノードのうち 2 つをグループから外すと全てのデータノードは 2 つのパリティグループに属すようになる。これが overlapped 7-neighbors である (図 4.19)。overlapped 7-neighbors も 1 種類のパリティグループを重ね合わせるもので、各パリティグループは 7 つのノードから構成される。この形状を用いた場合トーラスネットワークの一辺のノード数は 2 の倍数でなければならない。overlapped 7-neighbors のデータノードからパリティノードまでの平均距離は 1.33 である。overlapped 7-neighbors の冗長率は $1/4$ である。overlapped 7-neighbors でも MPN は使えない。

表 4.7: overlapped 9-neighbors を用いた DR-net と各方式の MTTF の比較

構成	ディスク数	冗長率	MTTF
RAID レベル 1	24	(1/2)	$0.30/\lambda$
RAID レベル 6, EVENODD	20	1/5	$0.23/\lambda$
DR-net(O9N)	25	1/5	$0.21/\lambda$
RAID レベル 3 〜 5	25	1/5	$0.16/\lambda$
DR-net(O9N-MPG)	25	1/5	$0.15/\lambda$
RAID レベル 6, EVENODD(D.P.)	25	1/5	$0.13/\lambda$
RAID レベル 3 〜 5(D.P.)	25	1/5	$0.082/\lambda$
RAID レベル 0	25	(0)	$0.0400/\lambda$

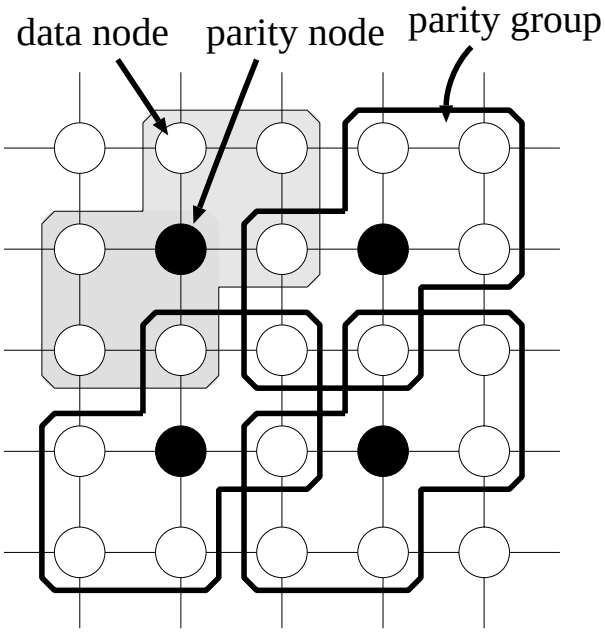

図 4.19: overlapped 7-neighbors

表 4.8: overlapped 7-neighbors と各方式の MTTF の比較

構成	ディスク数	冗長率	MTTF
DR-net(O7N)	36	1/4	$0.239/\lambda$
RAID レベル 1	36	(1/2)	$0.238/\lambda$
RAID レベル 6, EVENODD	32	1/4	$0.21/\lambda$
DR-net(O7N-MPG)	36	1/4	$0.19/\lambda$
RAID レベル 3 〜 5	36	1/4	$0.14/\lambda$
RAID レベル 6, EVENODD(D.P.)	36	1/4	$0.086/\lambda$
RAID レベル 3 〜 5(D.P.)	36	1/4	$0.056/\lambda$
RAID レベル 0	36	(0)	$0.028/\lambda$

overlapped 7-neighbors を用いた DR-net の MTTF を 6×6 の構成について求めると、

$$MTTF_{\text{O7N}} = \frac{0.24}{\lambda}$$
$$MTTF_{\text{O7N(MPG)}} = \frac{0.19}{\lambda}$$

となる。同程度の冗長率を持つ他の方式との比較を表 4.8に示す。declustered parity でないときの RAID レベル 3 〜 5 は $N_G = 5, G = 5$、また RAID レベル 6 では $N_G = 2, G = 1\,0$とした。

各パリティを別々のディスクに保持した結果、ディスク 9 台分の冗長情報を持つ構成で 9 個のディスク故障に耐えられることがわかる。

4.5 関連研究

DR-net と同様に、2 つのディスク故障に対してデータの喪失を防ぐシステムがいくつか提案されている。RAID レベル 6[5] は、Reed-Solomon 符号を用いることにより各パリティグループ内で 2 つまでのディスク故障に対応できる。しかし、従来の RAID レベル 5 とは異なり、XOR 演算器以外にも Reed-solomon 符号の計算のためのハードウェアが必要である。DR-net では、パリティを冗長情報に用いているため、XOR 演算器のみで計算が可能で

ある。[10] では、ディスクアレイに適用するさまざまな符号に関して論じられている。特に 2d-パリティ符号は XOR 演算のみで計算され、2 つのディスク故障に対するデータ喪失を防ぐ多くのシステムの基本的な考え方を示している。しかし、この方法ではパリティを格納する専用ディスクが必要であり、パリティ更新負荷の集中による性能低下が避けられない。DR-net では、パリティを全てのディスクで分散保持することにより性能低下を防ぐことが可能である。EVENODD[1][2][3] は XOR のみで計算されるパリティを冗長情報として利用し、全てのディスクにパリティを分散させることが可能である。EVENODD ではパリティグループを構成するディスク数は素数に限定される。これらのディスクは必ずしも実際のディスクでなくてもよく、0 のみを格納する仮想的なディスクを追加することで、ディスク数を素数とみなすことができる。しかし、そのような操作を加えることにより、各ディスクブロックが属すパリティグループや、そのパリティグループ内の他のディスクブロックを特定するためのコントローラのロジックがやや複雑になる。また、EVENODD ではパリティストライプ内の特別な位置に属すブロックを更新する際には、付加的な XOR 演算が必要であるため、DR-net よりも多くの XOR 演算を必要とする。[24][25] では、RM2 と呼ばれる手法について述べられている。RM2 では EVENODD でみられた付加的な XOR 演算は必要なく、1 パリティグループ内のディスク数も比較的柔軟に選択できる。しかし、各パリティグループに属すディスクのブロック番号がディスク毎に変化するため、各パリティグループに属すブロックとその物理的な位置の対応が複雑であり、故障ディスク内のデータの再構築を行なう場合などに必要なブロック位置を高速に求めるための対応表を記憶しておくことが必要と思われる。DR-net では、各パリティグループに属すブロックのブロック番号は同一であるため、そのような対応表は必要ない。また、RM2 を用いるシステム構成の信頼性については言及されておらず、RAID6 との相対的な信頼性などは不明である。

4.6 まとめ

冗長率をそろえた 25 台のディスクノードを持つ構成で具体的に MTTF を比較した結果、従来のパリティグループを用いた DR-net は、いずれも RAID レベル 3 〜 5 よりも高い信頼性を有し、特にパリティを分散させなかった場合には RAID レベル 6 やレベル 1 をしのぐ非常に高信頼なシステムであることを示した。RAID レベル 6 との差は、RAID レベル 6 の MTTF を基準として約 13%であり、有意な信頼性向上であるといえる。また、パリティ

を分散した場合には、MPN の方が信頼性は高いが、MPG と MPN の間にはそれほど大きな差はないことが明らかになった。

一般的な規模での信頼性に関しては、MTTF による比較は困難であるが、十字型のパリティグループを用いる構成についてマスク可能な最大故障数について考察した結果、ほぼ規模に比例して増加することが分かった。パリティを固定した場合では、RAID レベル 6 よりもマスク数は大きくなり、MPN で分散した場合でもほぼ RAID レベル 6 と同数の故障がマスクできることが示された。各構成方式のカバリッジの大小関係が規模の変化にともなって大きく変化しないとすれば、パリティを固定した DR-net は一般的に RAID レベル 6 よりも高い信頼性を持ち、パリティを分散した場合でも RAID レベル 3 〜 5 よりも十分に高い信頼性を持つと考えられる。

信頼性と記憶効率の関係については、DR-net でも記憶効率の柔軟な選択が可能であることを示し、いくつかの構成例を示した。それらの信頼性を検討した結果、overlapped 7-neighbors を用いた DR-net では、RAID レベル 1 や同じ冗長率を持つ RAID レベル 6 よりも高い信頼性があり、記憶効率を改善した場合でも RAID レベル 3 〜 5 よりも高信頼で RAID レベル 6 に匹敵する DR-net の構成が可能であることが示された。また、MPG でパリティを分散した場合も、ディスク台数の違いを考慮すれば RAID レベル 6 と同程度の信頼性が得られると思われる。

2 つのパリティ分散方式を比較すると、4.2.3で述べたように、一般に MPG よりも MPN を用いる方が信頼性が高い。しかし第 3章で述べたように、MPN ではパリティ更新などの際に行なわれるデータノードとパリティノード間の通信の平均距離が MPG に比べて大きいため、更新アクセスが多い場合には MPN は性能面で MPG よりも劣る可能性がある。動作方式の選択にあたっては、信頼性だけでなく性能についても考慮する必要があるが、これらの動作方式の性能面への影響については、次章で議論する。

第 5 章

動作方式と性能の関係

DR-net の 2 つのパリティ分散方式は、どちらも全てのディスクに均等にパリティを分散配置するが、前章で示したようにその信頼性に相違点があった。同様に性能面でも、パリティグループ内でのノード間の平均通信距離の違いから、更新処理やデータ再構築処理において 2 つの動作方式には違いがあると思われる。本章では、小規模な実験システムを用いた実測結果に基づき、第 3章で述べた DR-net の各動作方式が性能に与える影響について検証する。

5.1 実験システム

本節では、実験に用いたシステムのハードウェアおよびソフトウェアの実装について述べる。

5.1.1 ハードウェア

実験システムは、25 台のディスクノードと 5 台のインタフェースノードから構成され、5×5 の 2 次元トーラスネットワークで結合されている。各ノード間通信は store and forward で転送される。ディスクノードは、次のような構成である [31][35][38]。

コントローラ 通信、演算用コントローラとしてトランスピュータ T805、ディスクコントローラとして CORAL HPT04 SCSI-2 TRAM

ディスク Quantum Go-Drive 120S($2.5''$ 120MB SCSI ハードディスクドライブ)

表 5.1: ディスクノード性能

XOR 性能	1.25MB/sec
通信バンド幅	887KB/sec
平均シーク時間	17ms
平均回転待ち時間	8.3ms
最大転送速度	4MBytes/sec

これらと共にネットワークコンフィギュレーション用 C004 リンクスイッチチップや電源装置をケースに収めたものが合計 25 ユニット接続されている。また、インタフェースノードにも T805 が使用され、そのうちの 1 台はシステム制御用として IMS B300 TCPlink を介してイーサネット経由で外部のワークステーションと接続されている。残りの 4 台のインタフェースノードは変換回路からシリアルリンク (RS422) を介してパーソナルコンピュータのプラネトロン社製リンクアダプタボード LA-AT-F に接続されている。

今回用いた実験機の、インタフェースノード /外部環境間の転送速度は比較的低速 (847KB/sec) であり、性能面で大きなオーバヘッドとなることが考えられる。また、ノード間通信に関しても store and forward で転送されるため、転送路が長い場合には大きな影響が考えられる。そこで、アクセス要求をインタフェースノードではなく内部のディスクノードから自分自身へ発行することにより実験機に付随するオーバヘッドを極力排除した条件で、各種の DR-net の動作方式を比較する。

表 5.1にコントローラおよびディスクの諸パラメータを示す。

5.1.2 ソフトウェア

コントローラ用のプログラムは occam2 で記述した。コントローラにおける通信, ディスクアクセス, XOR 演算を並列に行なうために各機能はスレッドとして実装される。プログラム全体のサイズはコメントを除いて約 5500 行、コンパイル後のバイナリサイズは評価用コードおよびデバッグ情報を含めて約 187KB で、比較的小さなプログラムで実装されている。

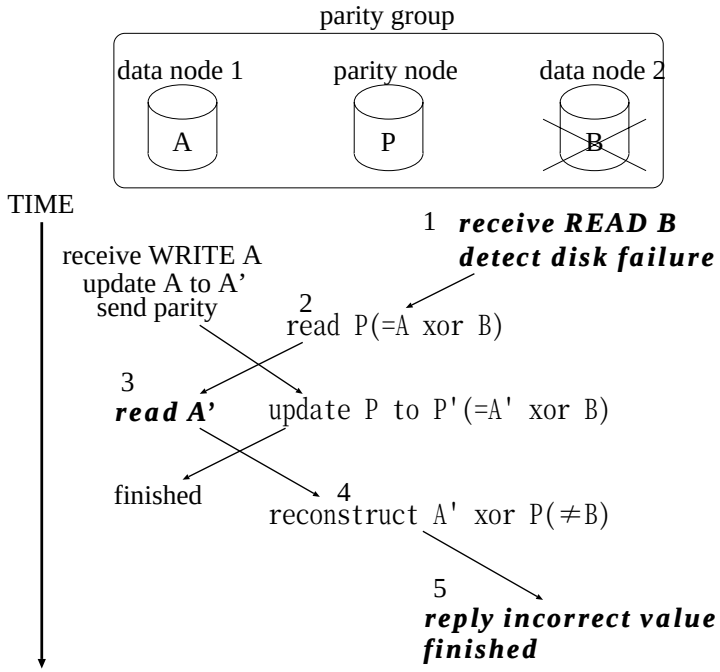

図 5.1: 誤ったデータ読み出しの例

デッドロック

各ノードに送られたユーザからのアクセス要求やデータ再構築などのための他ノードからの要求は、ノード内のスレッドによって処理されるため、あるノードへ要求を出してもノードのスレッドが他の処理中であればその要求は処理されずに、ブロックされる。複数の要求が、互いに相手のスレッドを待ち合う状況になるとデッドロックが発生することになる。

これを避けるためには十分な数のスレッドを用意すればよいが、トランスピュータではスレッドにともなう ALT ループの増加により、スレッド数の 2 乗に比例して性能が低下することが分かっている。そのため、実験機では要求処理のスレッドは 1 つだけ用意して、処理の途中で明示的にコンテキストを切替えることで複数の要求を並行して処理している。

データとパリティの整合性

DR-net では、書き込みやデータの再構築では 1 つの要求を複数のノードで処理するため、パリティグループ内のデータとパリティの整合性の維持に注意しなければならない。ディスク故障が存在する場合、誤った読み出し (図 5.1) や更新 (図 5.2) が発生する可能性があ

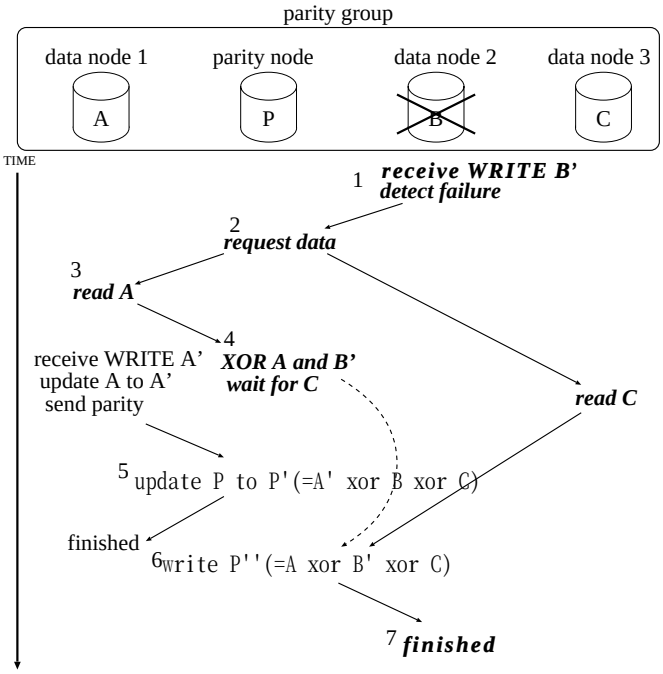

図 5.2: 誤ったパリティ更新の例

る。通常、RAID などでは整合性の維持にはロックによる排他制御が用いられるが、DR-net では処理に関わるノードが故障ディスクの位置によって異なるためあらかじめ必要なロックを特定できず、ロック取得の順序によっては並行して処理される要求との間でロック待ちによるデッドロックが発生する。また、ロック取得のために付加的に発生する通信などのコスト増加により、本来整合性の問題が生じない無故障時の性能にも大きな影響を及ぼすことになる。

この問題を解決するため、データやパリティブロック内の一部にバージョン番号を記載し、ディスクブロックのバージョン管理を行なう。図 5.3のように各データブロックにバージョン番号をつけ、パリティブロックには生成に使用したデータブロックのバージョン番号を列挙する。データブロックを更新する際にはバージョン番号をインクリメントし、パリティのバージョン番号もパリティ生成に用いたデータブロックのバージョン番号に更新する。データとパリティを併用するような処理ではそれぞれのバージョン番号を照合し、不一致の場合にはデータやパリティを読み直して処理をやり直す。

図 5.1の例では、図中 4 のデータ再構築の際に 2 で読み出したパリティブロック P に記載されているデータブロック A のバージョン番号と 3 で読み出した A' のバージョン番号

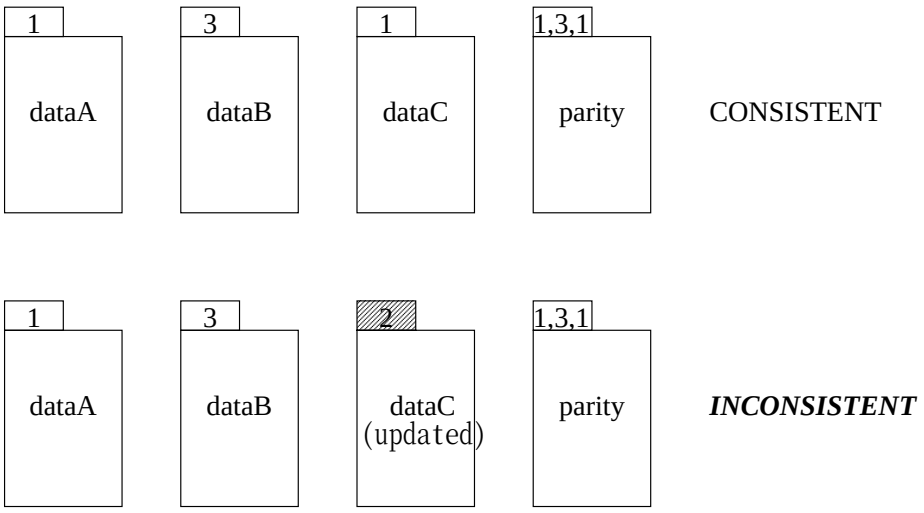

図 5.3: ディスクブロックのバージョン番号による不整合の検出

が異なるため、正しい結果が得られないことが検出される。図 5.2のような更新処理では、新しいパリティを書き込む前に、新しいパリティの生成に使用したデータのバージョン番号を、更新前のパリティのバージョン番号と比較する。

図 5.2の例では、6 で新しいパリティを書き込む前に、5 で書き込まれたパリティとバージョン番号を照合する。その結果、本来更新する B 以外にも A のバージョン番号も異なっていることが分かる。新しく書き込むデータ以外のデータブロックのバージョン番号が一致しなければ、パリティの生成に必要ないずれかのブロックが更新されていることがわかるので、もう一度データブロックを読み直す。この場合の照合に用いる古いパリティのバージョン番号は、パリティノードでパリティ更新のための処理が始まってから最後に新しいパリティを書き込むまでの間 (図中 2 から 6 の間) に、同じパリティブロックに対する書き込みをノード内で監視し、書き込みが行なわれた場合にはそのバージョン番号を覚えておくことにより得られる。複数回の書き込みがあった場合には、最新のものだけを覚えておけばよい。もし監視している間に一度も当該ブロックに書き込みがなければ、バージョン番号の照合は必要ない。

まとめると、バージョン管理で付加される性能面でのコストは表 5.2のようになる。ディスク故障が存在しない場合の付加は小さく、バージョン不一致の際の再読み出しは高コストであるが、発生確率が極めて低いことを考慮すれば十分容認できる。故障ディスクに対する書き込みでのパリティブロックのアクセス監視やバージョン番号の記憶についても、

表 5.2: 各処理に付加されるコスト

アクセス処理	付加されるコスト
読み出し	なし
書き込み	データおよびパリティのバージョン番号更新
読み出し (故障)	データおよびパリティのバージョン番号照合。
	不一致の場合の再読み出し。
書き込み (故障)	データおよびパリティのバージョン番号照合。
	当該ブロックに対する書き込みの監視とそのバージョン番号の記憶。
	不一致の場合の再読み出し。

処理負荷や必要な記憶空間はほとんど無視できる範囲と考えられる。

5.2 ディスク故障が存在しないとき

本節では、ディスク故障が存在しないときの DR-net の性能を測定し、パリティ分散保持方式と性能の関係について検証する。

5.2.1 DR-net 内部でのスループットのモデル

スループットを支配する最も大きな要因はディスクアクセスであると考えられる。他にも、ノード間通信やパリティ計算の処理が考えられるが、これらは前述の条件下ではディスクアクセス時間に比べて十分小さく、またディスクアクセスと並列に処理されると考えれば無視できる。そこで、ディスクアクセスの負荷に注目し、単体ディスク、パリティ固定保持の DR-net、パリティ分散保持の DR-net におけるそれぞれの平均スループット算出のモデルを示す。ただし、各ディスクブロックのアクセス頻度が均等であるとする。

単体ディスク

単体ディスクにおける、1ブロックの読み出し, 書き込みの所要時間をそれぞれ t_R, t_W(ms)、また、1 回のアクセスデータ長を L(Bytes) とすると単体ディスクの読み出し, 書き込みのス

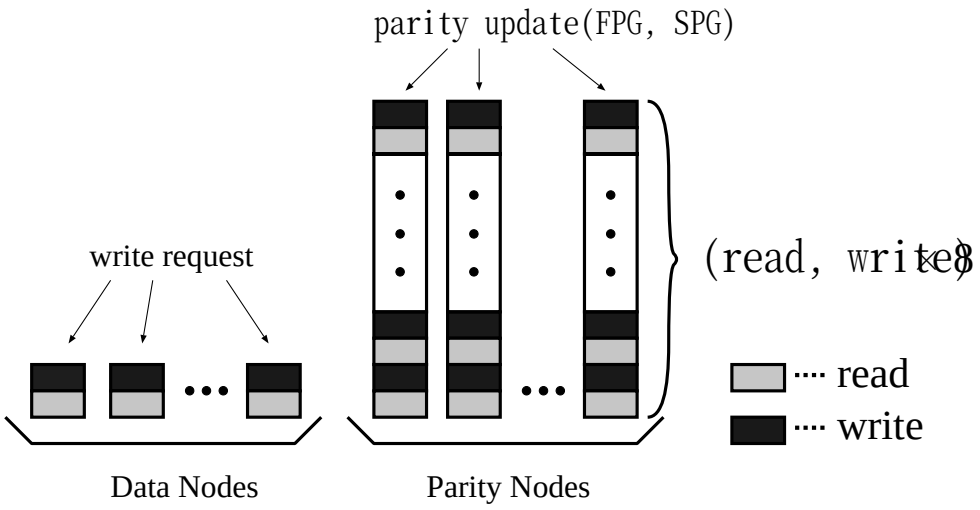

図 5.4: パリティ固定保持方式の書き込み処理

ループット T_{SR}, T_{SW}(KB/s) はそれぞれ

$$T_{SR} = \frac{L}{t_R} \tag{5.1}$$

$$T_{SW} = \frac{L}{t_W} \tag{52}$$

となる。今回用いた実験システムでは、ディスクアクセスの際に内部でバッファリング処理があるため、単純な実測では t_R, t_W が計れない。そこで、スループットを計測することにより、上式から 1 ブロックあたりの平均アクセス所要時間 t_R, t_W を求める。これらの値は、以下のモデル式で使用される。

パリティ固定保持方式

読み出しはパリティディスクを除いたすべてのディスクで並列に処理される。データディスク数を N_D とすると、読み出しスループット T_{FR} は

$$T_{FR} = \frac{N_D L}{t_R + OH_{FR}} \tag{53}$$

となる。式中の OH_{FR} はディスクアクセス以外のオーバヘッドである。

書き込みではパリティの更新が伴う。すべてのデータノードに 1 回の書き込み要求が出されると、式 (3.1) からわかるように、データディスクでは旧データの読み出しおよび新データの書き込みのために read, write が 1 回ずつ発生する。一方、パリティディスクには

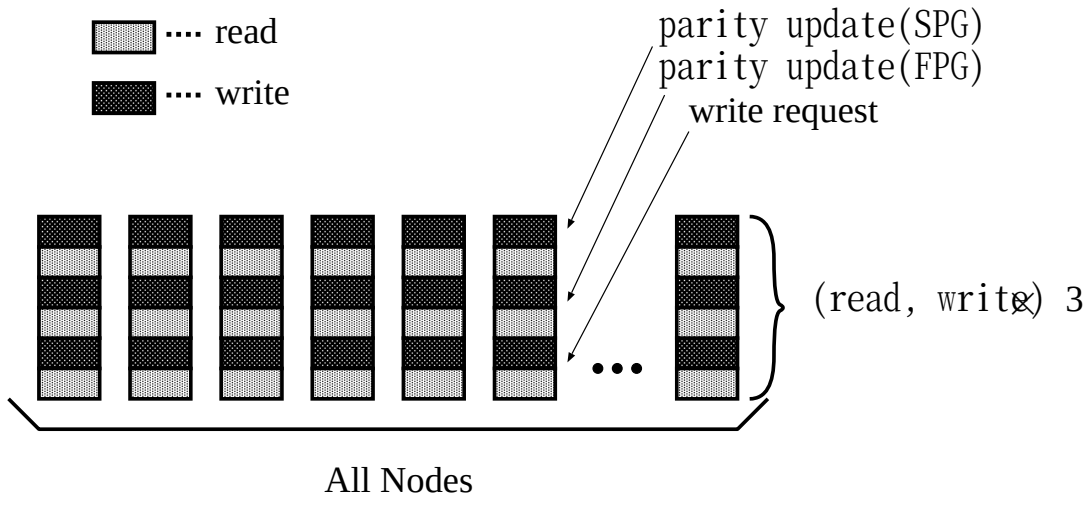

図 5.5: パリティ分散保持方式の書き込み処理

8 つのデータノードからパリティ更新要求が集中するため、旧パリティの読み出しおよび新パリティの書き込みのための read, write が 8 回ずつ発生する。従って、パリティディスクの処理のスループットが支配的となる。8 回の read, write を行なう間に N_D 回の要求を処理すると考えられるから (図 5.4)、

$$T_{FW} = \frac{N_D L}{8(t_R + t_W) + OH_{FW}} \quad (54)$$

パリティ分散保持方式

読み出しはシステムの全ディスクで並列に処理される。全ディスク数を N とすると、

$$T_{DR} = \frac{NL}{t_R + OH_{DR}} \quad (5.5)$$

となる。

書き込みの際のパリティ更新は分散して行なわれる。1 回の書き込み操作につき、データの更新, FPG のパリティ更新, SPG のパリティ更新が必要であるから、データノードおよびパリティノードにおいて合計で 3 回の read, write が発生する。全てのディスクに対して 1 回の書き込み要求が出され、パリティ更新がすべてのディスクで均等に分散されるとすると、各ディスクで read, write が 3 回ずつ発生する。3 回の read, write を行なう間に N 回の要求を処理すると考えられるから (図 5.5)、

$$T_{DW} = \frac{NL}{3(t_R + t_W) + OH_{DW}} \quad (56)$$

となる。

5.2.2 性能評価実験

ディスク故障がない場合の読み出しおよび書き込み性能についてについて実験を行なった。システムに対し 1000 回のディスクアクセスを要求し、アクセスはブロック単位で行なった。アクセスは全てのノードに均等に分散され、アクセスするブロックはランダムである。アクセス要求は、宛先のノードが受信可能の状態になると直ちに発行される。ディスク故障が存在する場合は、アクセス時にコントローラにより直ちに検出される。各アクセス要求の結果は、各要求の処理が全て終了してから返される。

各構成での読み出し、書き込みのレスポンスタイムおよびスループットの測定結果をそれぞれ図 5.6, 5.8, 5.7, 5.9に示す。比較のため、スループットでは各構成のモデル式でオーバヘッドを 0 とした場合の値も示してある。

レスポンスタイム

読み出しのレスポンスタイムは、ほぼ単体ディスクと同等の値を示している。読み出しでは、各アクセス要求の処理は、その宛先のディスクノードのみで処理され、ノード間で影響を及ぼすことがないためにオーバヘッドが小さくなっている。一方、書き込みでは、単体ディスクでの書き込みとは異なり、データノード、パリティノードの双方で read-modify-write が必要となるため、レスポンスタイムは、単体ディスクと比較して大きく増加している。しかし、MPG, MPN では、パリティを分散しなかった場合と比較して約 1/2 のレスポンスタイムが得られ、パリティ更新の負荷が分散された効果が表れている。

スループット

図 5.8から、読み出しに関しては実測値とモデル値が比較的一致することがわかる。読み出しでは、ディスク台数の増加に見合う性能向上が達成されていることが確認できる。

一方、書き込みスループットはモデル値よりもかなり低い値となっている (図 5.9)。モデルと比較してオーバヘッドを算出すると、図 5.10のようになりオーバヘッドの大きさがブロックサイズに比例していることがわかる。このことから、オーバヘッドの原因としてはパリティ計算やノード間通信の時間がディスクアクセスで隠蔽されていないことやデータのコピーや転送など、データ長に関係するソフトウェアオーバヘッドが考えられる。

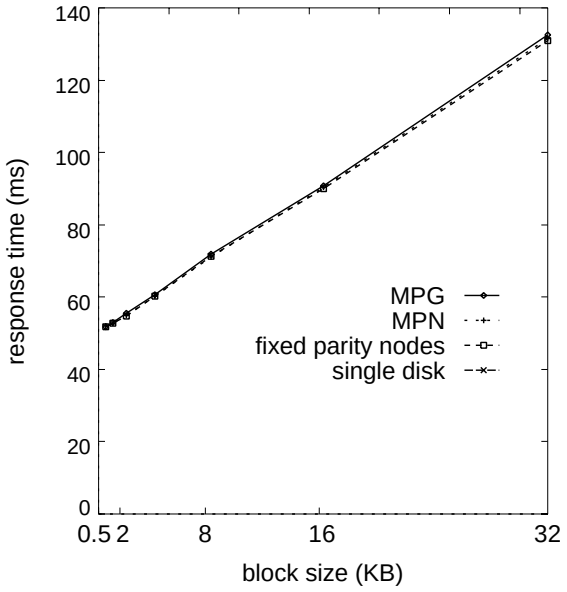

図 5.6: 各方式の読み出しレスポンスタイム

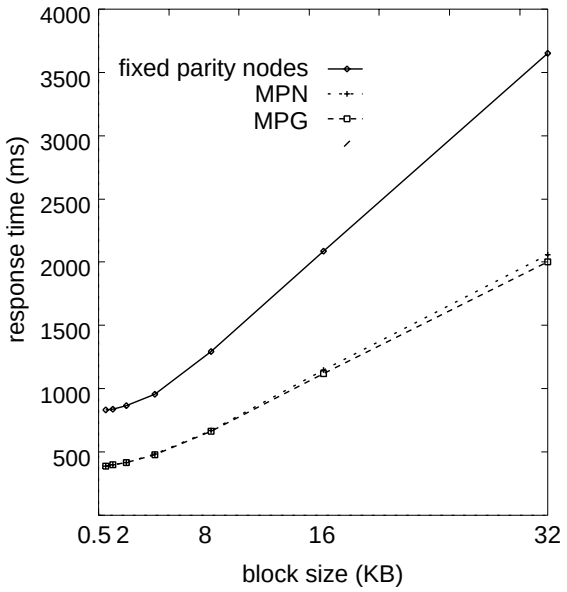

図 5.7: 各方式の書き込みレスポンスタイム

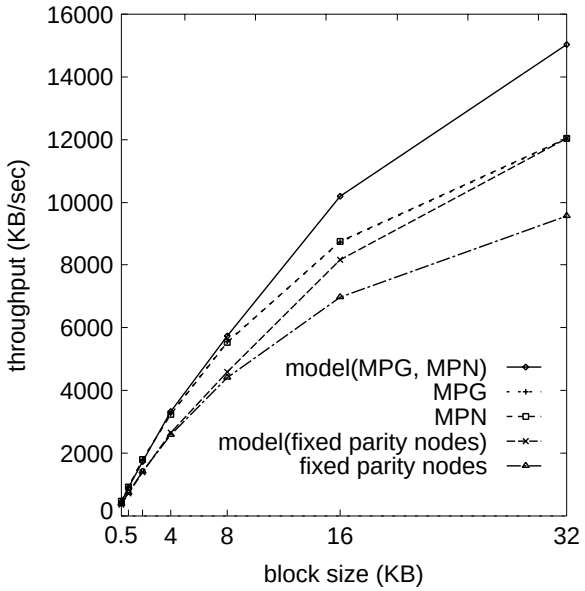

図 5.8: 各方式の読み出しスループット

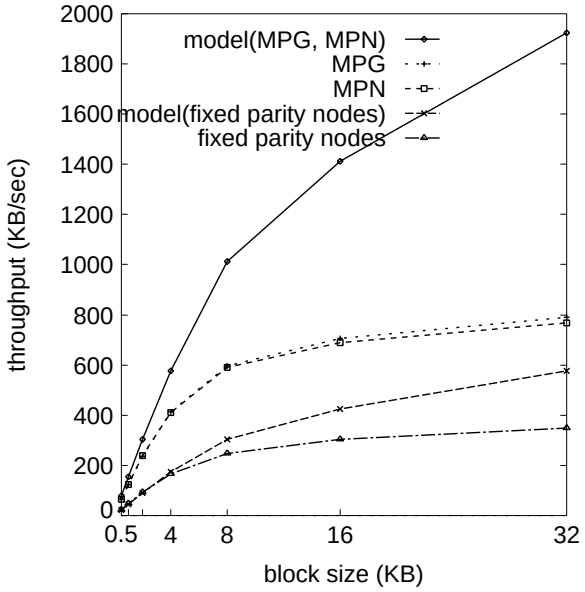

図 5.9: 各方式の書き込みスループット (ディスク故障なし)

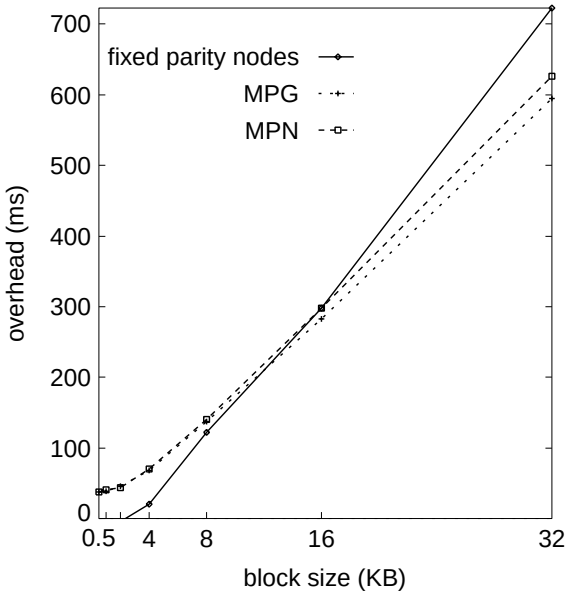

図 5.10: 各構成の書き込みオーバヘッド (ディスク故障なし)

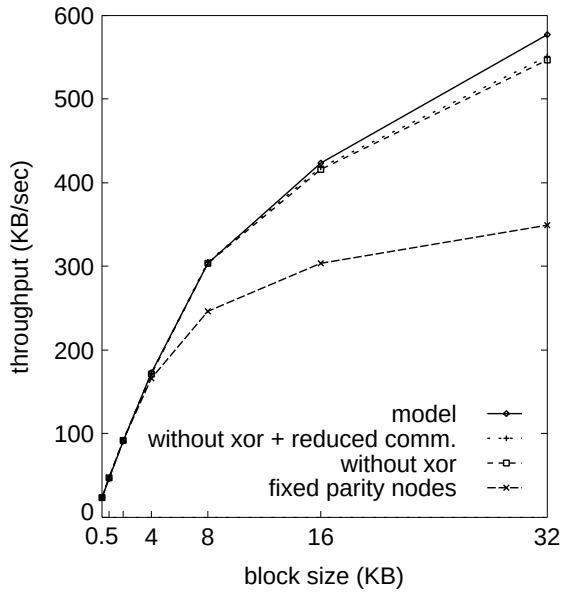

図 5.11: パリティ計算、通信量を削減した固定パリティノードの書き込みスループット (ディスク故障なし)

表 5.3: パリティ分散によるスループットの向上率

ブロックサイズ (KB)	MPG	MPN
0.5	2.69	2.65
2	2.62	2.61
8	2.53	2.47
16	2.48	2.44
32	2.44	2.40

原因を確かめるため、パリティ計算を省略あるいは通信時に送るデータ量を削減した実験を行なった (図 5.11)。この結果、パリティ計算を省略することによりモデルに近い値が得られた。一方、通信量を削減した場合には、あまり性能は向上しなかった。このことから、モデルと実験結果の不一致の原因は、パリティ計算に伴う処理がディスクアクセスにオーバラップされない実験システムの問題と考えられる。

パリティ保持方式の比較

読み出しスループットの改善は、データを保持するディスク台数の増加により、パリティを分散しない場合に比べ、より高並列な動作が可能となったことによる。書き込みでは、それに加えてパリティ更新負荷の分散がスループットを向上させている。各ブロック長でのパリティ分散をしない場合に対する向上率は表 5.3のようになり、パリティ分散保持の効果が確認できる。一方、モデルから書き込みスループットの向上率 T_{DW}/T_{FW} を求めると

$$\frac{T_{DW}}{T_{FW}} = \frac{NL}{3(t_R + t_W) + OH_{DW}} \times \frac{8(t_R + t_W) + OH_{FW}}{N_D L}$$

OH_{FW}, OH_{DW} を 0 とすると、実験システムでは $N = 25, N_D = 20$であるから、

$$\frac{T_{DW}}{T_{FW}} = \frac{25}{3} \times \frac{8}{20} = \frac{10}{3}$$

従って、スループットは $10/3$ 倍になると期待できる。この値と実測値との差は書き込みにおけるオーバヘッドが 0 でないことによると思われる。

また、MPG と MPN を比較すると、ブロックサイズが大きくなると MPG の方が若干高いスループット、短いレスポンスタイムを示していることがわかる。MPG と MPN でデー

タノードとパリティノードの平均距離を計算すると、FPG についてはそれぞれ 1, 1.6 で、SPG についてはそれぞれ 2, 2.4 となる [37]。そのため、通信距離が短い MPG の方が store and forward による通信時間やメモリ内でのデータ転送時間が短く、また並行して処理される要求間でのリンク上の通信衝突も少ないため、性能面で有利であることが結果に表れていると考えられる。

5.3 ディスク故障が存在するとき

実験における故障ディスク数は 0 〜 7 とした。ディスク故障が存在する場合にはシステムに対しデータ再構築のための負荷が加わる。しかし、同じ故障数でも故障ノードの位置により負荷の重さやノード間での負荷の分散の度合が異なるため、解析的に性能をモデル化することは困難である。ここでは、再構築不可能な故障パタンでの性能については考慮せず、各パリティ保持方式で再構築が可能なパタンについて実験を行なった。その際、全ての位置パタンについて実験することは困難であるため、ここでは各故障ディスク数について 10 の再構築可能な位置パタンを選び、それらの結果の重み付き平均値を評価結果とした。パタンの選択は、各パタンについて全てのノードをアクセスしたときにシステムで発生するディスクアクセス数を調べ、その値が近いものは 1 つにまとめるようにし、各故障数毎に最大 10 のパタンで代表させた。その代表パタンについて実験を行ない、それぞれまとめたパタンの数だけ重みをつけて全体の平均をとった。また、アクセス要求が出されたノードのディスクが故障していた場合、故障は直ちに検出されシステムはデータ再構築等の故障時動作を行なう。その他の条件は、ディスク故障が存在しない場合と同様である。

5.3.1 再構築戦略の比較

1 つのデータブロックは FPG, SPG の 2 つのパリティグループに属し、その再構築の際にはどちらでも利用できる。従って、故障ディスク内のデータの読み出しには、第 3章で説明したように、LRS, ERS の 2 つの戦略が考えられる。書き込みの際には、FPG, SPG の 2 つのパリティブロックを両方とも更新しなければならないため、LRS, ERS ともに同じ動作となる。

図 5.12, 5.13は、それぞれ MPG を用いてパリティを分散させた場合のレスポンスタイムおよびスループットである。どちらも、ERS より LRS の方が良い結果が得られた。また、

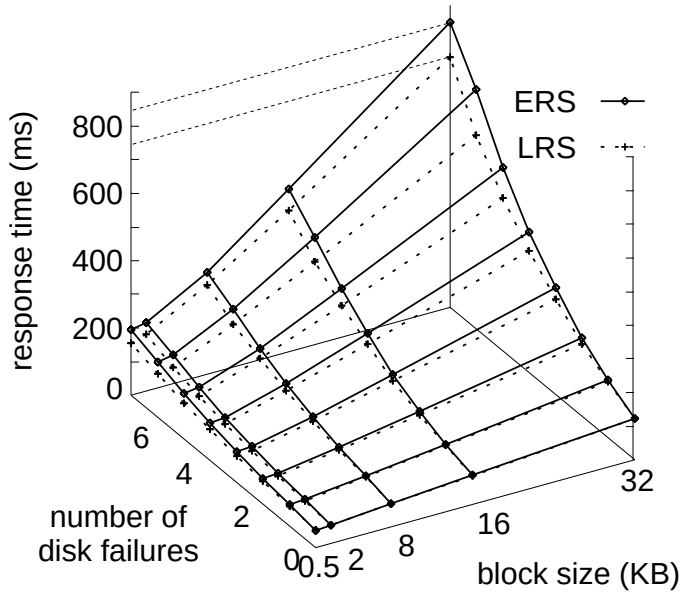

図 5.12: LRS, ERS のレスポンスタイムの比較 (MPG での読み出し)

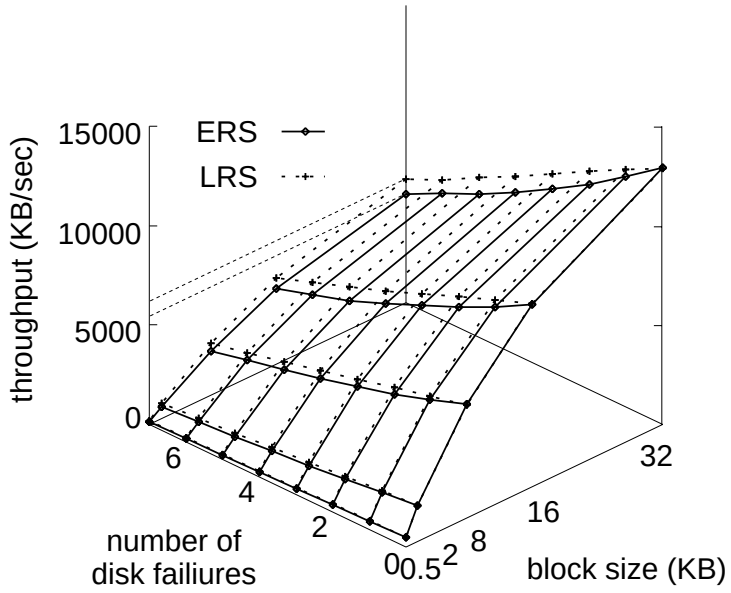

図 5.13: LRS, ERS のスループットの比較 (MPG での読み出し)

パリティ固定保持, MPN の場合も同様の結果が得られている。

ERS は 2 つのパリティグループで同時に再構築を開始し、早く得られた結果を利用する。しかし、これにより ERS では LRS に比べて多くのノードでディスクアクセスが行なわれ、それぞれのノードの負荷を増加する。その結果、故障していないディスクに対する読み出し要求の処理にも影響を与え、LRS に比べてレスポンスタイムやスループットが悪化していると考えられる。

5.3.2 パリティ分散方式の比較

図 5.14, 5.15は、LRS を用いた読み出しのレスポンスタイムおよびスループットである。MPG と MPN を比較すると、故障ディスクがあるときの読み出しでは MPN の方が良い性能を示している。ERS を用いた場合も同様の結果が得られている。

考えられる原因の一つとして、故障ディスクのデータを再構築するために必要なディスクアクセス数が考えられる。同じ故障パタンでも、各フェーズで MPG と MPN のパリティグループの配置が異なるため、ある故障ディスクの再構築処理に関わるディスクノードも異なる。1 つの故障パタンで各フェーズ毎に全てのディスクに 1 回ずつアクセス要求を出したときにシステム全体で合計何回のディスクアクセスが起きたか数え、各故障数毎に平均すると図 5.16のようになる。これらの図からわかるように、MPG は常に MPN よりも多くのディスクアクセスが必要であり、このことが原因で MPG よりも MPN の方が良い結果を出したと考えられる。

同様に、書き込みに関しても MPG の方が多くのディスクアクセスが必要である。しかし、故障ディスクがないときの書き込みでは MPG の方が MPN よりも良い性能を示しているため (図 5.9)、故障数が少ない場合の多少のアクセス数増加による影響が打ち消され、MPG が優位を保つ。しかし、故障数が多くなるとアクセス数増加の影響が顕著になり MPN の方が良い結果を出すと考えられる (図 5.17, 5.18)。

5.3.3 パリティ固定保持方式での書き込み

図 5.19に故障が存在する場合のパリティ固定保持方式での書き込みスループットの変化を示す。パリティを分散させた場合には、故障ディスク数の増加にともなって書き込みのスループットも緩やかに低下していくが、パリティ固定保持方式では、スループットが故障数に関して単調に低下せず、故障数が 6 以下ではほとんど性能が劣化していないことが分

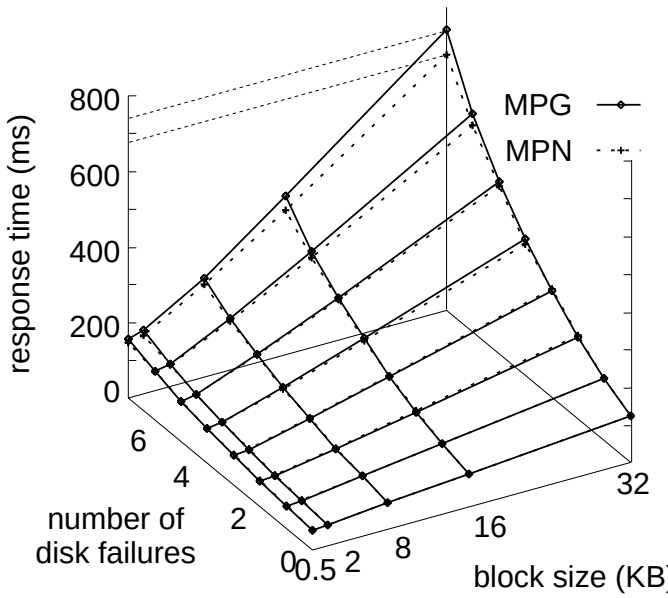

図 5.14: MPG, MPN のレスポンスタイムの比較 (LRS での読み出し)

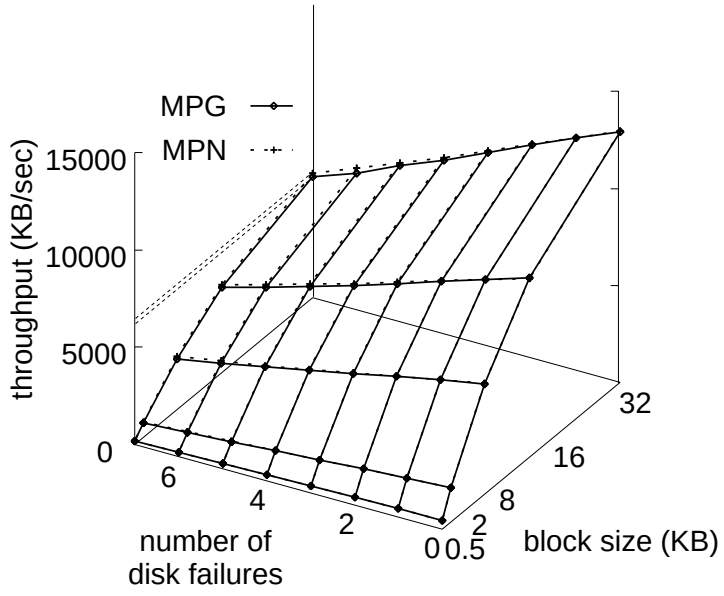

図 5.15: MPG, MPN のスループットの比較 (LRS での読み出し)

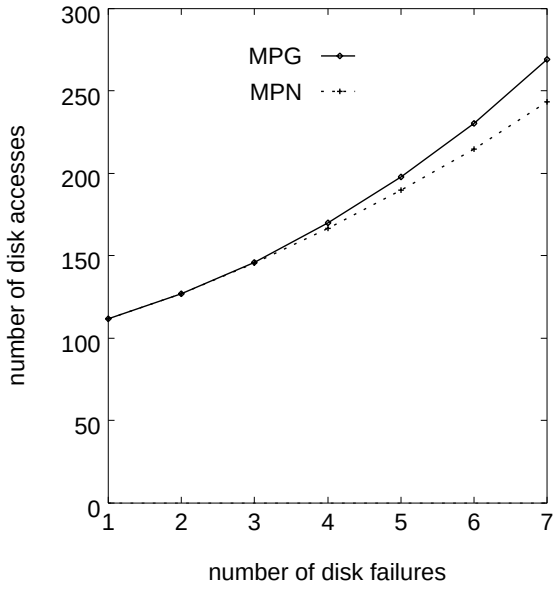

図 5.16: LRS を用いた再構築に必要なディスクアクセス数の比較

かる。正常なディスクへの書き込みに伴うパリティ更新は式 (3.1) で計算されるが、式 (3.1) の計算には旧パリティが必要なことから、パリティノードにおいて 1 回の読み出し（旧パリティの読み出し）と 1 回の書き込み（新パリティの書き込み）が必要である。一方、故障ディスクへの書き込み要求に伴うパリティ更新は式 (3.2) で計算されるため、パリティノードにおいて旧パリティの読み出し処理は行なわれず、パリティノードの負荷は軽減される。その反面、式 (3.2) では書き込むデータノードと同じパリティグループに属す他の 3 つのデータノードのデータを必要とするため、他のデータノードの負荷が増大する。従って、書き込みを要求されたディスクが故障していると、

- 負荷の重いパリティノードにおける負荷の軽減
- 負荷の軽い他のデータノードにおける負荷の増大

という相反する 2 つの要因が考えられる。これらの要因により、パリティ固定保持方式での書き込みは、故障ディスク数が少ないときには性能のネックとなっていたパリティノードの負荷軽減により、故障ディスク数が多くなると他のデータノードの負荷増大により再びスループットが減少すると考えられる。

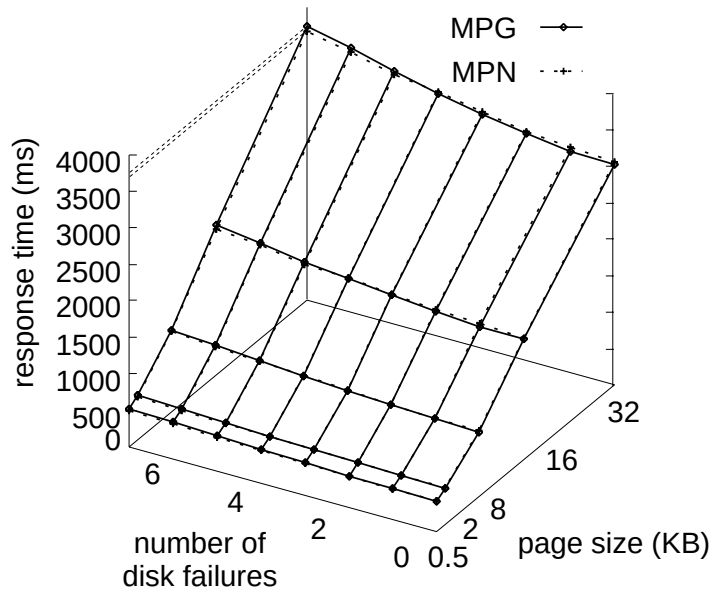

図 5.17: MPG, MPN のレスポンスタイムの比較 (書き込み)

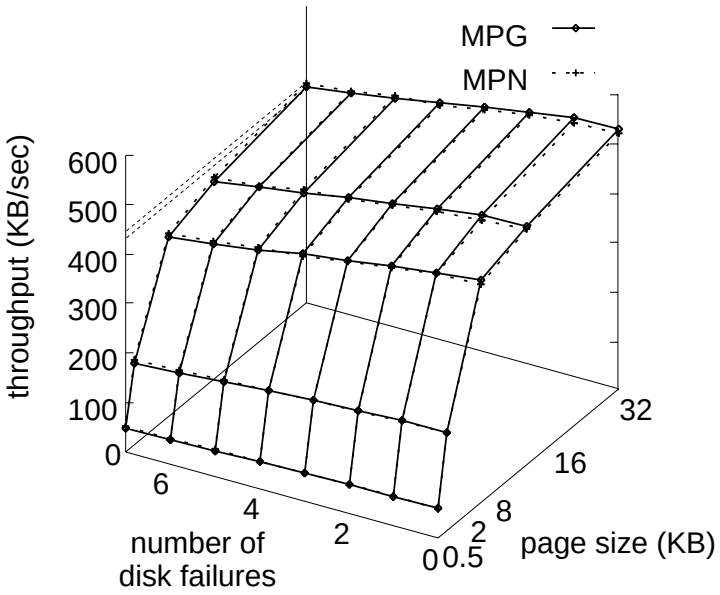

図 5.18: MPG, MPN のスループットの比較 (書き込み)

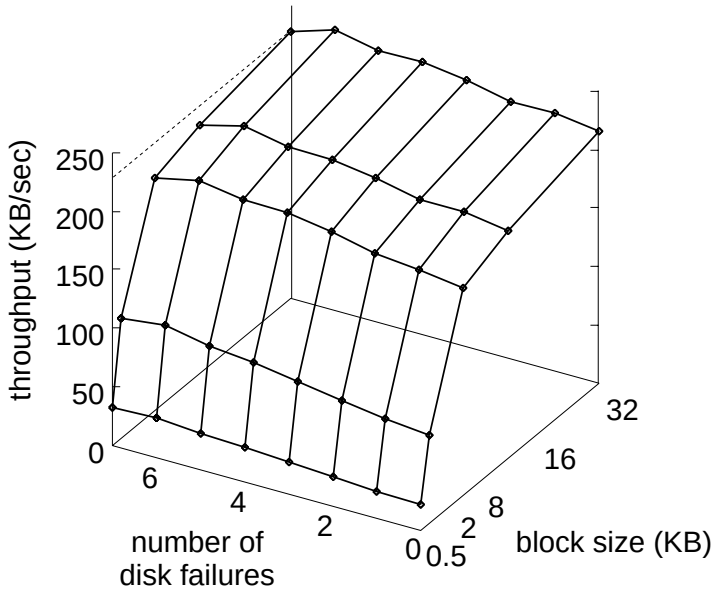

図 5.19: パリティ固定保持の書き込みスループット (ディスク故障あり)

5.4 関連研究

本章で用いた実験システムと同様に、トランスピュータを用いて実装したシステムが存在する。TickerTAIP[4] はディスクを持つノードや外部環境と接続されるノードに T805 を利用し、全体をネットワークで結合したプロトタイプシステムを実装している。このプロトタイプを利用して、設計の指針や必要な制御用プログラムのサイズなどが得られている。TickerTAIP では制御の分散に関していくつかの構成が考えられるが、集中制御方式のみが実装された。本章で用いた実験機は複数のインタフェースノードがそれぞれ独立に動作する DR-net として実装されている点で違いがある。なお、本章で示したオーバヘッドと同様に、TickerTAIP のプロトタイプでもディスクアクセスと計算がオーバラップされないことが報告されている。

データとパリティの整合性維持に関しては、[12][17][25] などでロックによる排他制御について言及されている。[12] や [17] では、ディスクアクセス処理の過程の一部としてロックの取得が挙げられているが、RAID レベル 4 や 5 などが対象であるため処理に関わるノードがあらかじめ確定できる。従って、ロックによる排他制御を用いても本章で示したような問題は生じない。一方、RM2[25] では DR-net と類似した冗長情報が利用されており、整

合性の維持に関しても DR-net と同様の問題が生じる。RM2 ではパリティストライプ毎にロックがあり、ブロックの更新処理の際にはブロックが属すパリティストライプ全体をロックする。排他制御の実装に関しては述べられていないが、単一のコントローラにより全てのパリティストライプを集中管理することを想定していると思われる。しかし、ディスクの容量や台数の増加にともない、パリティストライプ数が大きく増加した場合には単一のコントローラがボトルネックとなることが考えられる。DR-net で用いるディスクブロックのバージョン管理では、システムの規模に関わらず処理コストが低く、性能に影響を及ぼさない。

パリティの分散方法と性能の関係については、[14] でさまざまな配置法について議論されているが、ネットワーク通信を考慮しない 1 つまたは 2 つのパリティグループに対する複数ブロックのまとまったアクセスを対象としている点で、本章の議論とは異なる。

また、[18] では、ディスク故障が存在するシステムでの性能が論じられているが、主に予備領域へのデータ再構築時の性能を取り上げている。しかし、ディスク台数が多くなった場合には故障の頻度も増加し、また故障ディスクの交換がそれほど頻繁に行なわれないような環境では予備領域を使い果たすことも考えられる。そのような環境を想定する場合には、本章で示したような予備領域がない場合の性能が重要となる。

5.5 まとめ

さまざまなパリティ分散方式やデータ再構築戦略での DR-net の性能について、システム内部のスループットのモデルを示し、実験機を用いた実測結果を示した。

モデルおよび実験結果から、書き込み性能に関しては DR-net では 1 つのデータ更新に対して 2 つのパリティ更新が伴うため、パリティを分散配置することが重要であることが確認された。パリティを分散した場合でも、読み出しに比べると性能が低いが、この問題については第 7章でとりあげる。

MPG と MPN の比較では、故障が存在しないときの書き込みではパリティグループ内のノード間距離の短い MPG の方がより効果的であることが確認できた。しかし、故障が存在するときは MPG でのデータ再構築には平均して MPN よりも多くのディスクアクセスが必要であり、MPN の方が良い性能を示す傾向があることがわかった。第 4章で示したように、MPN はもともと MPG よりも高い信頼性を持つが、データ再構築などの故障時の性能

が良いということは故障ディスク交換後のデータ再構築の高速化にもつながり、MPG と比べてさらに高い信頼性を持つと考えられる。

また、ディスク故障が存在する場合のパリティ固定保持方式の書き込み性能に関しては、故障ディスクの数が少ないときにはパリティノードの負荷が軽減されることから性能が改善されるが、故障ディスク数が多くなると他のデータノードへの負荷が増大し、再び性能が悪化することが示された。

2 つのデータ再構築戦略に関しては、いずれの場合も再構築の際に比較的少ないノードを必要とする LRS が有利であることがわかった。ただし、今回の結果は負荷の高い場合の特性を示しており、負荷が非常に低く、システム内でディスクアクセスがほとんど起きていないような場合には、ERS を用いた方が短いレスポンスタイムが得られる可能性がある。

本章では実験機に実装した特定の規模の DR-net を用いて、動作方式の違いが性能に与える影響を示したが、性能差の原因はシステムのノード数などに依存しないため、ここで示された性能面での相対的な関係は構成規模によらない一般的な性質と考えることができる。ただし、システムの絶対的な性能はディスクノード数やインタフェースノード数に依存して変化することが予想される。システムの構成規模の変化と性能の関係については次章で議論する。

第 6 章

構成規模や通信バンド幅の性能への影響

前章では、特定の規模の実験機を用いて動作方式間の性能の違いを示したが、システムの構成規模が大きくすることにより、より多くのディスクを並列にアクセスすることが考えられる。また、そのような条件では、第 3章で述べた DR-net の特徴を生かした性能向上が期待される。そこで、本章ではシステム構成の規模や通信性能が性能に与える影響に注目して DR-net の性能を解析する。

システム内のディスクノード数, インタフェースノード数, 通信バンド幅などを変化させたときのシミュレーション結果から、DR-net の最大の特徴である内部ネットワークと複数のインタフェースの性能面での効果について考察する。また比較のため、3 種類のバスシステムについても同様の実験による評価結果を示す。

6.1 シミュレーションモデル

6.1.1 システム構成

実験で評価する DR-net は、第 5章で用いた実験機と同様に、2 次元トーラスネットワーク上に第 3章で示したパリティグループを配置する構成を対象とする。

比較対象としてのバスシステムには、RAID-II[6] で採用されているような階層構造を用いる。ここでは、階層化レベルが異なる 3 種類の構成を用いる。これらのバスシステムは、DR-net と同様、FPG と SPG がディスクノードにマッピングされており、バスとネットワークの違いを除けば、DR-net と全く同じ動作をする。

1. フラットなバスシステム

   図 6.1のように一本のバスに全てのインタフェースノードとディスクノードが結合した単純な構成である。各通信の経路長は、全てのシステム中で最も短いが、全ての通信が単一のバスを通して行なわれるため、ディスク数が大きくなるとバス上での通信衝突が増加すると思われる。

2. 2 階層バスシステム

   図 6.2のようにバスを 2 段に階層化したシステムである。このシステムでは、ディスクノードは 25 台毎に分割され、それぞれのグループが一本のローカルなバスを持つ。ローカルなバスはバスコントローラで上位のバスに結合されている。グループ内のディスクノードは、ローカルなバスを介して互いに通信する。グループ間に跨ったパリティグループはないため、パリティ更新などのノード間通信は全てローカルなバス上でのみ行なわれ、通信の局所性が保たれている。従って、上位のバスはディスクノードとインタフェースノードの間の通信のみで使用される。

3. 3 階層バスシステム

   前述の 2 段階層システムをさらに 1 段階層化したものである (図 6.3)。各下層バスは 5 台のディスクノードによって共有され、それぞれ FPG を形成している。下層バスはバスコントローラを介して中層バスに接続されている。1 本の中層バスにつき 5 本の下層バスが接続されている。データ更新の際には、FPG のパリティ更新のための通信は下層バスだけで行なわれる。SPG のパリティ更新には下層バスと中層のバスが使用される。この構成では、2 階層のシステムよりも通信負荷が分散されるが、インタフェースノードとディスクノードの通信には 2 つのコントローラを介することになる。

バスシステムにおけるインタフェースノードは、全て最上層のバスに結合されている。階層化による通信の分散は静的な局所性がある通信に対して有効であるが、多くのユーザからの発行されるアクセス要求には局所性がなく、また、あったとしても動的に変化するため、階層化による負荷の分散は困難である。従って、インタフェースノードを下層のバスに結合しても、ほとんどの通信は最上層のバスを経由して行なわれるため利点は少なく、返って通信の経路長が長くなることが予想される。ここでとりあげる階層バスシステムで

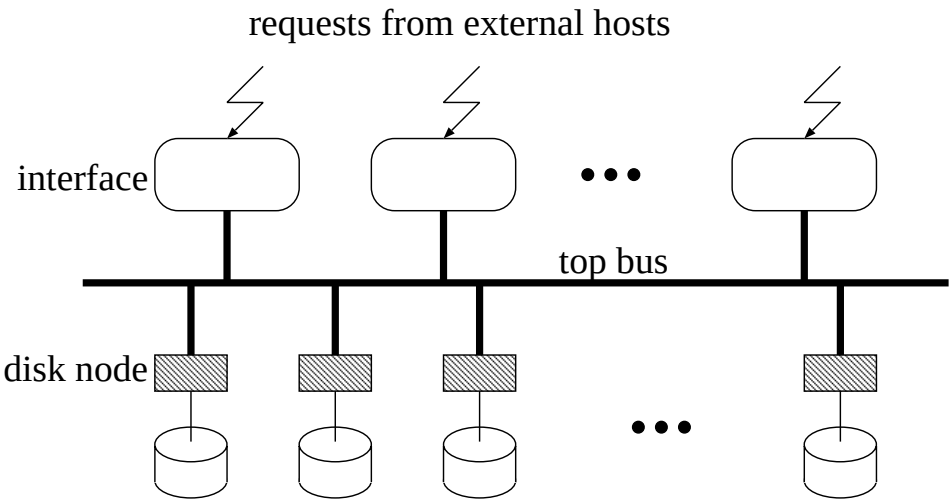

図 6.1: フラットなバスシステム

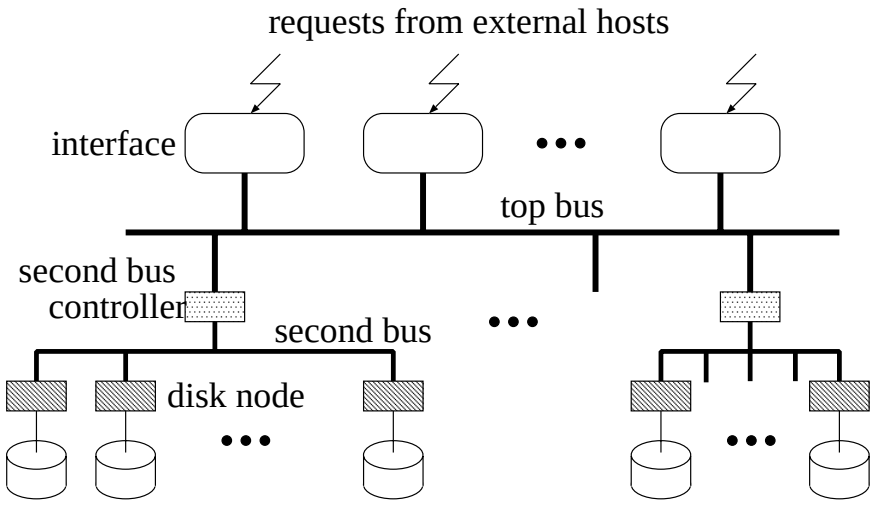

図 6.2: 2 階層バスシステム

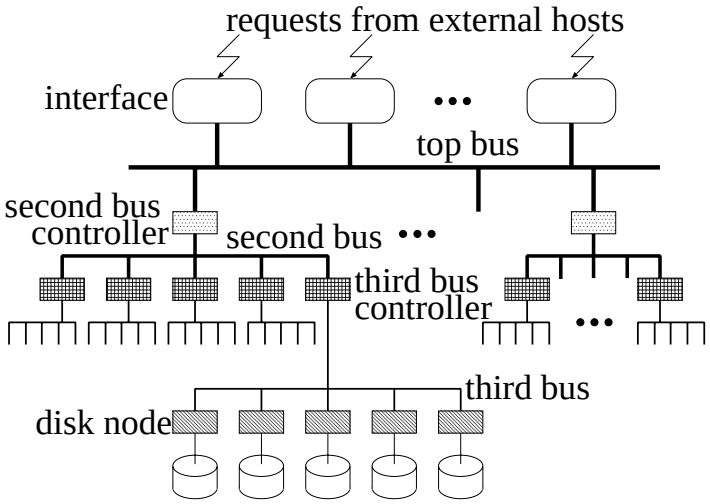

図 6.3: 3 階層バスシステム

は、パリティ更新のためのノード間通信を下層のバスに分散し、最上層のバスの負荷を下げる。

いずれのシステムでも、パリティブロックを MPN により分散配置し、書き込み性能の低下を防ぐ。MPN を用いる理由は、MPG によって動的にパリティグループの編成を変えてしまうと、パリティグループ内の通信の局所性が階層バスシステムで保たれないためである。

6.1.2 インタフェースノード

インタフェースノードは、READ および WRITE のアクセス要求を外部から受け取り、FIFO キューに入れる。アクセス要求は、キューの先頭に来てからシステム内部に発行される。各システム内には複数のインタフェースが存在し得るため、2 つ以上のインタフェースが同じディスクノードに並行して要求を送る可能性がある。ディスクノードが要求の受信に使用できるバッファや処理のためのスペースには限りがあるため、発行する要求がディスクノードに受信されることを保証するために、インタフェースノードはあらかじめ宛先のディスクノードに予約要求を送る。それに対して確認が返送されてから実際のアクセス要求を発行する。確認が返送されるまでは、インタフェースノードはブロックされる。インタフェースノードは、1 つの要求を発行したらその結果を待たずに、直ちに次にキュー

表 6.1: リンクおよびバスのパラメータ

	セットアップ時間 (μ sec)	バンド幅 (Mbits/sec)
DR-net リンク	5	80
上層バス	5	2000
中層バス	5	400
下層バス	5	80

の先頭に来る要求の処理を行なう。

6.1.3 アクセス要求

アクセス要求の宛先となるディスクノード、ディスクブロック、送り元インタフェースノードなどはランダムに選ばれる。外部のたくさんのホストと接続されていることを想定して、要求は途切れることなくインタフェースノードに供給される。1 回のシミュレーションで読み出し、あるいは書き込みの要求を 10000 回発行する。

6.1.4 リンクおよびバス

バスシステムでは、階層の異なるバスの間はバスコントローラによって接続される。バスコントローラが異なる階層間の通信を中継する際には、まず全てのデータを受信し、それから別の階層に送り出す。一方、DR-net のリンクは双方向で、データはワームホールルーティングで転送される。フリットサイズは 16 バイトである。ノード間通信を行なう際に経路上のリンクで通信衝突が起きる場合には、リンクが解放されるまで通信は開始されずにブロックされる。

表 6.1に、シミュレーションで用いたバスおよびリンクのパラメータを示す。DR-net のノード間リンクとして利用できるような point-to-point の接続を提供するネットワークとしては、Myrinet などを始めとして 100Mbits/sec をはるかに越える転送能力を持つものが珍しくないが、ネットワークを用いる効果をより明らかにするために、ここではそれほど高性能ではないネットワークを想定した。バスシステムでは、10Mbytes/sec の SCSI-2 を参考に下層バスの転送レートを設定し、中層バスは配下に結合されるディスクノード数に比例して下層バスの 5 倍の転送レートとした。上層バスについては、ディスクノード数に比例

して転送レートを設定すると非現実的な性能となってしまうため、同様に中層バスの 5 倍の転送レートとした。ただし、6.3.3節ではリンクやバスの性能を変化させた場合の性能についても示している。

6.1.5 ディスクノード

インタフェースノードは、前述のように、要求を発行する前に宛先のディスクノードに対して要求受信用バッファを予約する。ディスクノードは予約に対し、現在の要求受信数を調べ、受信することが可能であれば直ちに返答する。これ以上受信することができなければ、現在の要求の処理が完了してから返答する。実験では、1 つのディスクノードが受信可能な要求数は 3 とした。すなわち、各ノードには最大 3 つの要求を処理するために十分なメモリがある。

ディスクノード内の、ディスク、通信コントローラ、パリティ計算の XOR 演算器は並列に動作する。従って、ある要求のためにディスクアクセスしながら、別の要求のために XOR を計算したりすることができる。ディスクのパラメータは、将来の実用化を目指して現在ディスクメーカで具体的に検討されている 2.5 インチディスクのモデルを参考にした。これらのパラメータを表 6.2に示す。シークタイムは次の式で計算される [14]。

$$S_t = \begin{cases} S_t = A(x-1) + B\sqrt{x-1} + C & (x \geq 1\ ) \\ 0 & (x = 0\ ) \end{cases}$$

ここで、x はディスクヘッドが移動するシリンダ数で、A, B, C は単一シリンダシーク、フルストローク、平均シークタイムを満たすように選ばれる定数である。回転待ち時間は、0 から最大値までの間でランダムに選ばれる [29]。

6.2 アクセス要求処理の内訳

実験結果の検討に先立ち、インタフェースノードから発行される各アクセス要求の処理時間の内訳について考察する。

読み出しと書き込みのいずれの場合でも、処理の流れは次のようになる。

1. インタフェースノードが、要求を発行するノードに対してバッファの確保を要求し、受信/作業スペースを予約する。

表 6.2: ディスクノードのパラメータ

受信可能要求数	3
XOR 演算性能	10 Mbytes/sec
セクタサイズ	512 bytes
ブロックサイズ	32 sectors
セクタ数/トラック	90
トラック数/シリンダ	6
シリンダ数/ディスク	100000
単一シリンダシークタイム	0.5 ms
平均シークタイム	5 msec
フルストロークタイム	12 msec
最大回転待ち時間	4.3 msec
ディスクコントローラオーバヘッド	1 msec
ヘッド切替え時間	1 msec
転送レート	75 Mbytes/sec

2. 要求を宛先に転送する。

3. 宛先のディスクノードで読み出し、あるいは書き込みの処理を行なう。

4. 結果をインタフェースノードに返送する。

従って、アクセス要求の処理時間には次の 3 つが含まれる。

- 宛先のディスクノードの受信バッファが全て使用中であり、インタフェースノード内で待たされている時間 (転送のためのバスやリンクの待ち時間は含まない)
- インタフェースとディスクノード間の要求および結果の転送時間 (バスやリンクの待ち時間を含む)
- ディスクノードでの要求の処理時間 (ディスクアクセス、パリティ計算など)

これら 3 つが処理時間全体に占める割合は、システムの状況により異なる。例えば、多くのノードで既に 3 つの要求が受信されていれば、バッファの待ち時間は長くなる可能性が高い。また、要求が書き込みであれば、読み出しの場合よりもディスクアクセスなどの処理量が多いため、要求の処理時間が長くなる。

以後、ディスクノードのバッファ待ち時間、インタフェースとディスクノード間の転送時間、ディスクノードでの要求処理時間をそれぞれ「待ち時間」、「転送時間」、「処理時間」と略記する。

6.3 実験結果

各システムの性能を、各アクセス要求の平均レスポンスタイムと、システム全体のスループットで評価する。ここでのレスポンスタイムは、アクセス要求がインタフェースノードの FIFO キューの先頭に来た時刻と、要求の返答が返った時刻の時間差を表す。スループットはユーザによって直接アクセスされたデータ量と、最初の要求が発行されてから最後の返答が返るまでの時間から計算される。グラフに示されている値は、各条件に対して行なわれた 3 回の実験結果の平均である。また、いずれの場合もディスク故障が存在しない normal mode の結果である。

6.3.1 ディスクノード数

図 6.4, 6.5, 6.6, 6.7はディスクノード数を 100 から 3,000 まで変化させたときの結果である。この実験では、インタフェースノード数は 40 である。

一般に、ディスクノード数が増えると要求の宛先が分散され、複数の要求が同じノードに発行される確率が下がるため、待ち時間は短くなる。また、それによって要求の発行レートが上がるため、スループットは向上する。

実験結果から、DR-net がどのバスシステムよりも増加したディスクノードを有効に利用し、スループット、レスポンスタイムの両面で高い性能を示していることがわかる。バスシステムのスループットが飽和している原因を考えると、DR-net の結果を見れば分かるようにディスクノードの処理能力にはまだ余裕があることから、上層のバスでの通信がボトルネックとなっていることは明らかである。

バスシステムの読み出しのレスポンスタイムは、ノード数の増加にともない一度は減少しているが、その後再び増加している。これは次のように理解できる。

1. まず、ディスクノード数の増加により待ち時間が減少し、要求発行レートも上がる。これは、レスポンスタイムの減少とスループットの向上が示している。

2. さらにノード数が増えると待ち時間は減少するが、システム内で多くの要求が並行して処理されるため、各要求間で通信に必要なバスの獲得競争が激しくなる。通信しようとするときにバスが使用中であれば、バスが解放されるまで処理はブロックされる。

3. これにより転送時間が増大し、レスポンスタイムの悪化につながる。また、要求の発行自体にもバスが必要なため、発行レートが上がらずスループットは飽和する。

バスシステムでは、待ち時間の減少よりも転送時間の増加の方が、レスポンスタイムの支配的要因となっている。DR-net では通信は分散されるため転送時間の増加よりも待ち時間の減少の方が支配的となる。

一方、書き込みのレスポンスタイムではバスシステムでもノード数に伴う増加が見られない。これは、書き込みでは上記の要因に加えて、ノードの増加にともなって処理時間が減少するためである。書き込みではパリティ更新が必要であることから、1 つの要求当たり 3 つのノードで 6 回のディスクアクセスを必要とする。従って、ディスクノード数が少ない場合には各要求間で互いにディスクを中心とする資源の獲得競争が発生し、処理時間

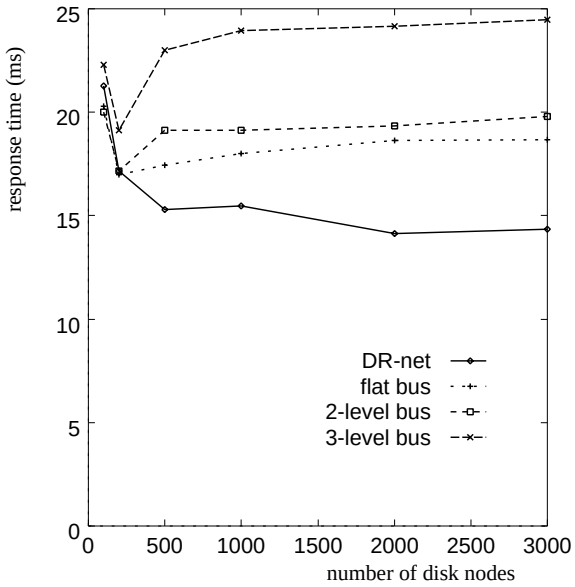

図 6.4: 読み出しのレスポンスタイム (インタフェース数=40)

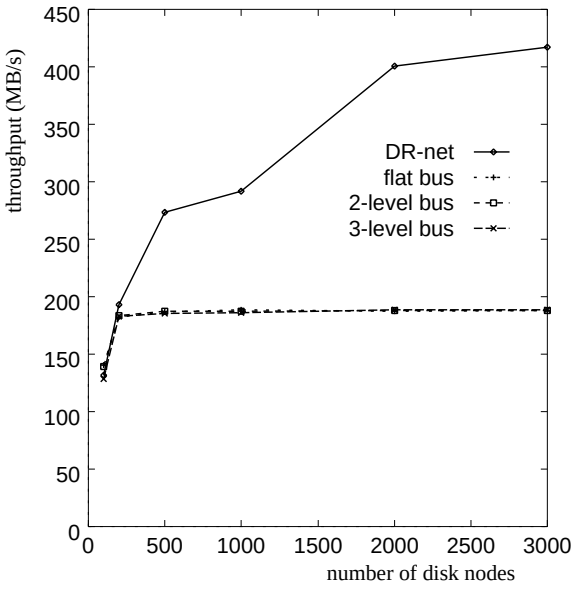

図 6.5: 読み出しのスループット (インタフェース数=40)

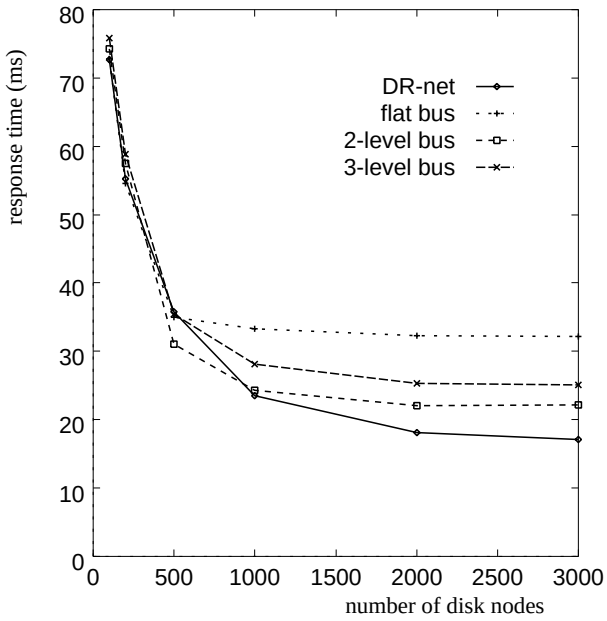

図 6.6: 書き込みのレスポンスタイム (インタフェース数=40)

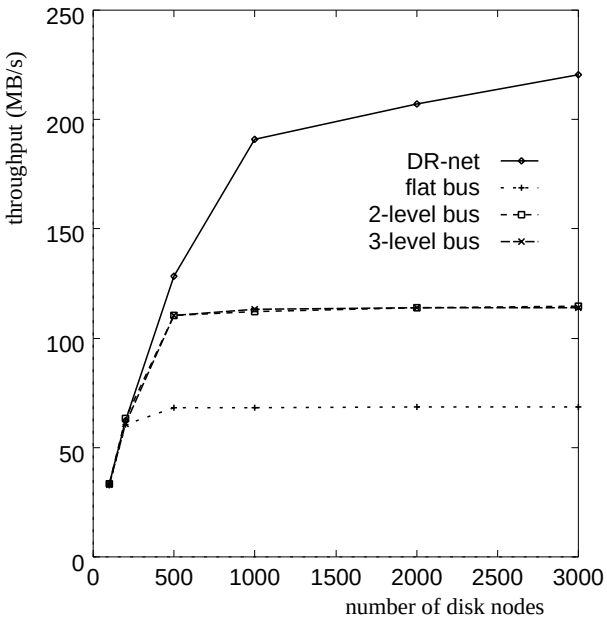

図 6.7: 書き込みのスループット (インタフェース数=40)

が長くなる。しかし、ディスクノード数が増加すれば負荷は分散され、資源獲得競争の緩和により処理時間が減少する。書き込みでは処理時間が支配的であるため、転送時間の増大よりも処理時間の減少がレスポンスタイムに表れていると考えられる。

6.3.2 インタフェースノード数

図 6.8, 6.9, 6.10, 6.11は、インタフェースノード数を変化させたときの読み出しの結果である。ディスクノード数は 1,000 である。

インタフェースノード数が増加すると、単位時間あたりより多くの要求が発行されるため、スループットは向上する。しかし、並行に処理される要求数の増加が処理時間や転送時間の増大を招き、レスポンスタイムは悪化する。

結果から、DR-net はインタフェースノード数が大きいときにバスシステムよりも高い性能を示している。バスシステムでは、ディスクノード数が増加した場合と同様に、バスがボトルネックとなってスループットが飽和する。

図 6.8から、DR-net の読み出しレスポンスはインタフェースノード数の増加にともなって一度改善され、その後再び悪化することが分かる。これは、インタフェースノード数が小さい場合には、通信がインタフェースノードの周辺リンクに集中し、ボトルネックとなるからである。インタフェースノード数が 1 のときの読み出しスループットは約 16MB/s であり、インタフェースノードの 2 本のリンクの転送能力 80Mbits/s×2 のかなりの部分を使っていることからわかる。これにより、インタフェースノード数が小さいときには読み出しのスループットも低く抑えられている。一方、書き込みでは処理時間が大きく要求の発行レートが低いため、リンクがボトルネックとなるほどスループットは上がらない。従って、レスポンスタイムは単調に増加する。

これまでの結果では、2 つの階層バスシステムの違いはあまり見られない。上層のバス性能が支配的であることから、3 階層のシステムでもスループットは改善されず、2 つのシステムの違いはインタフェースノードとディスクノードの間の通信が経由するコントローラ数の違いにより、レスポンスタイムに表れている。

6.3.3 通信バンド幅

図 6.12, 6.13, 6.14, 6.15は、通信バンド幅を変化させたときの性能の変化の様子である。グラフの横軸は、表 6.1の値との比を表している。つまり、横軸の目盛 2 は各バスと DR-net

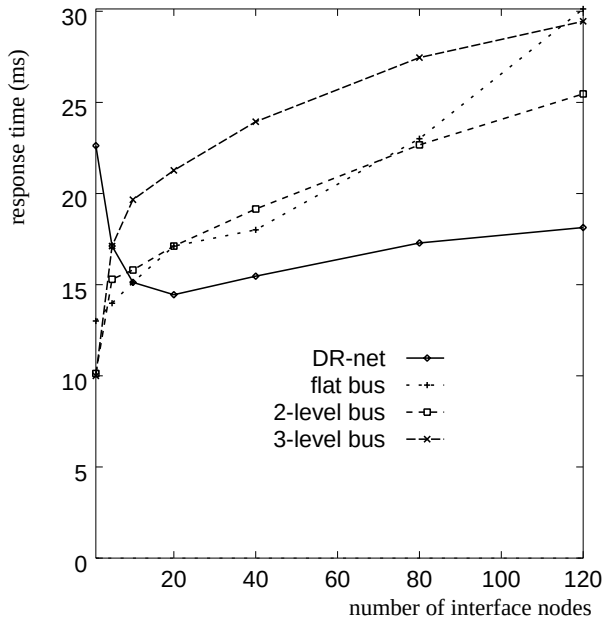

図 6.8: 読み出しのレスポンスタイム (ディスクノード数=1,000)

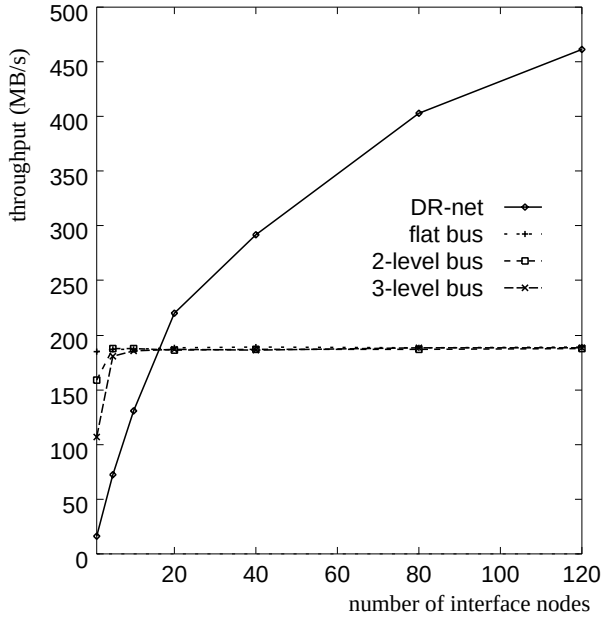

図 6.9: 読み出しのスループット (ディスクノード数=1,000)

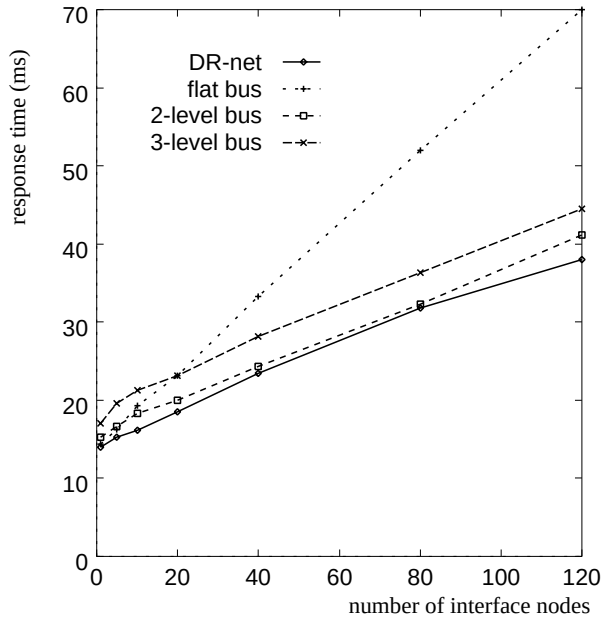

図 6.10: 書き込みのレスポンスタイム (ディスクノード数=1,000)

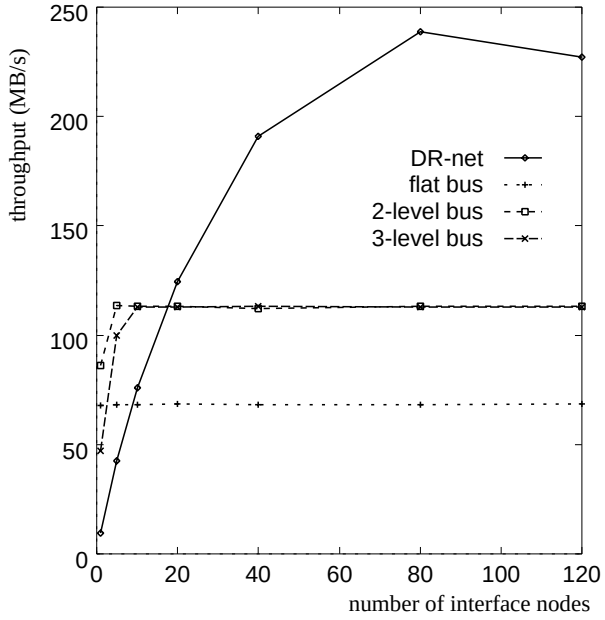

図 6.11: 書き込みのスループット (ディスクノード数=1,000)

のリンクのバンド幅がそれぞれ 4000, 800, 160, 160Mbits/sec であることを表している。シミュレータの都合上、バスシステムの横軸は 5 までであるが、全体の変化の傾向をとらえる上では問題ない。ディスクノード数は 1,000、インタフェースノード数は 40 である。

読み出しのレスポンスタイムでは、DR-net も 3 つのバスシステムも通信の衝突が緩和されてレスポンスタイムが減少している。また、スループットに関しても全てのシステムで向上しているが、DR-net の方が向上率は大きいことがわかる。図 6.13で、横軸が 2、つまり DR-net のリンクスピードが 160Mbps(= 20MB/s) に達すると、バスシステムでは 10Gbps 以上の性能のバスが必要となることがわかる。このようなバンド幅のバスは実現が難しく、やはり何らかのネットワークを用いることが必要であると思われる。

書き込みに関しては、どのシステムもほぼ単調にスループットが向上している。しかし、レスポンスタイムでは、DR-net、階層バス、フラットなバスの 3 つで違いが見られる。DR-net では、リンク性能の向上にともない転送時間が短縮され、要求の発行レートが上がる。従って、スループットが向上するが、システム内で並行に処理される要求数の増加により資源の獲得競争が激しくなるため処理時間が増大し、結果としてレスポンスタイムが増大する。階層バスシステムでも同様に処理時間が増大するが、バス性能が低いときには処理時間よりも転送時間の方が支配的と思われる。これは、図の横軸が 1 の条件では図 6.7, 6.11 などからわかるように通信ボトルネックによりスループットが飽和しているからである。従って、バス性能の向上に伴う転送時間の減少がレスポンスタイムを減少させ、さらにバス性能が上がると処理時間が増加してレスポンスタイムが増大する。一方、フラットなバスシステムでも、バス性能の向上により転送時間が減少し、発行レートが上がる。ところが、フラットなシステムではパリティ更新のためのノード間通信も同じバスを使用するため、処理中のアクセス数の増加にともないバス上の通信量が増加する。このため、階層バスの場合ほど発行レートが上がらず、システム内で並行して処理される要求数が比較的小さくなることから、資源待ちによるレスポンスタイムの増加が見られないと思われる。

6.3.4 システム規模

最後に、ディスクノード数/インタフェースノード数の比を 1000/40 に保ったままシステムの規模を変化させたときの結果を図 6.16, 6.17, 6.18, 6.19に示す。図の横軸は、ディスクノード数 1000、インタフェースノード数 40 のときを 1 とした規模の比である。これらの図に示される特性は、図 6.4から図 6.7と、図 6.8から図 6.11が組合わさったものと考えら

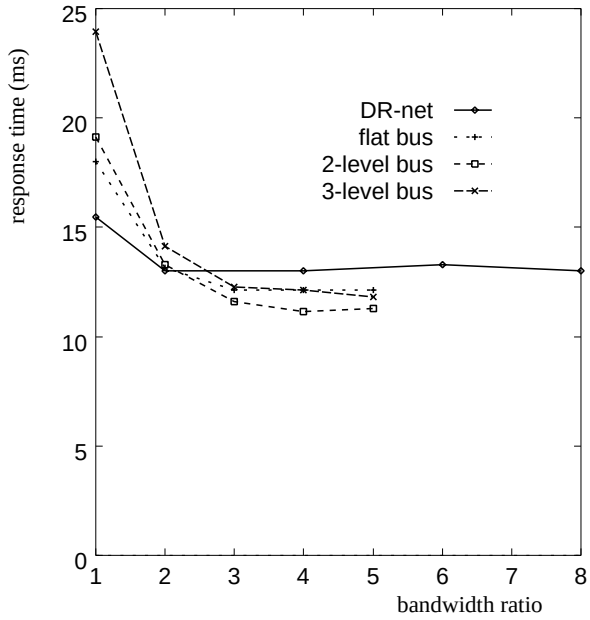

図 6.12: 読み出しのレスポンスタイム (ディスクノード数=1,000, インタフェースノード数=40)

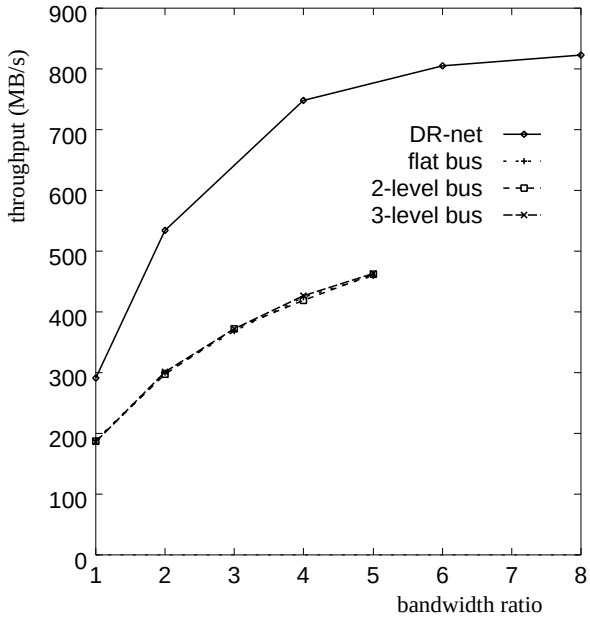

図 6.13: 読み出しのスループット (ディスクノード数=1,000, インタフェースノード数=40)

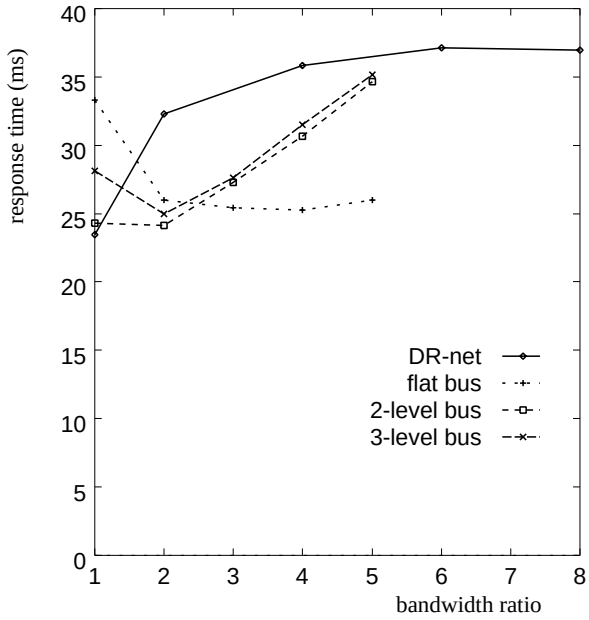

図 6.14: 書き込みのレスポンスタイム (ディスクノード数=1,000, インタフェースノード数=40)

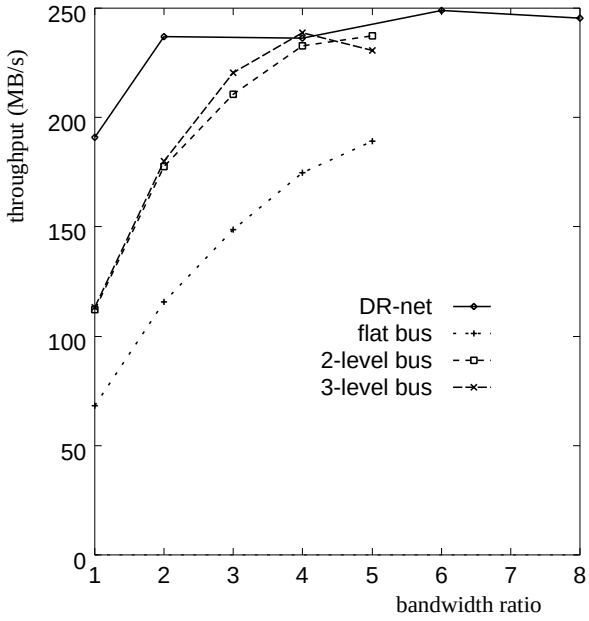

図 6.15: 書き込みのスループット (ディスクノード数=1,000, インタフェースノード数=40)

れる。DR-net はスループットおよびレスポンスタイムの双方で良いスケーラビリティを示し、大規模な構成に適していることが確認できる。

6.4 関連研究

二次記憶装置の内部にネットワークを導入した研究として TickerTAIP/DataMesh[4][32] がある。TickerTAIP は、ディスクを持つ各ノードがネットワークで結合され、複数のインタフェースノードが存在する点で DR-net と共通している。しかし、その性能評価ではネットワークによるノード間の完全結合を仮定しており、規模が大きくなった場合の構成については言及されていない。また、ネットワークトポロジを考慮したときのパリティグループの配置についても不明である。

[8][27] では、network-attached secure disk(NASD) と呼ばれる、セキュリティ管理などの機能を付加したディスクドライブを LAN に直結し、それらを統合してファイルシステムを構築する方法が提案されている。このアプローチは、クライアントとディスクドライブが LAN を経由して直接通信することによりデータ転送路からファイルサーバを外し、ネームスペースの管理とデータ転送の処理を分離することによってファイルサーバの負荷の軽減することを目的としている。各クライアントがデータ転送を並列に行なえる点で DR-net の複数のインタフェースノードとの類似点があるが、冗長情報の利用やその管理グループの構成法については述べられていない。

また、ネットワークで結合されたワークステーションクラスタを利用してファイルシステムを実現する研究がある [7][9][11][23]。いずれも、データや冗長情報を格納するストレージサーバとして機能するワークステーションと、ファイル管理やストライプ管理の機能を持つマネージャを高速ネットワークで結合し、全体で 1 つの二次記憶システムを構成する。これらのシステムは、概念的には DR-net と同様の構成であり、DR-net で用いられている手法をワークステーションクラスタに適用することが可能である。ただし、DR-net では個々のノードがディスクドライブに通信とパリティ計算機能を付加した簡単なものであるのに対し、ワークステーションクラスタでは各ノードに 1 台のワークステーションを用いるため、実装コストなどの面で違いがある。例えば、ベクトルマシンや汎用機の入出力サブシステムなど、ワークステーションクラスタとしての機能ではなく、純粋な二次記憶装置としての構成が要求されるような環境では、ワークステーションよりもより単体ディスクに

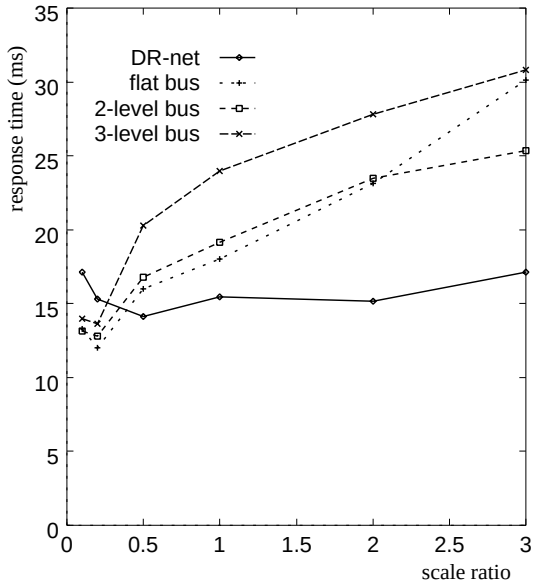

図 6.16: 読み出しのレスポンスタイム

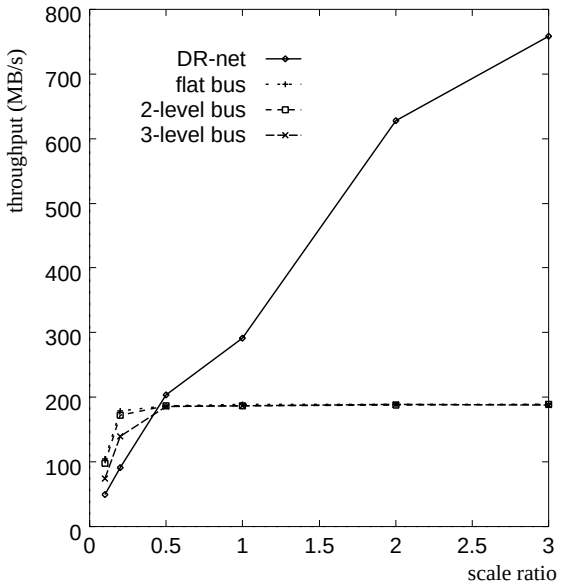

図 6.17: 読み出しのスループット

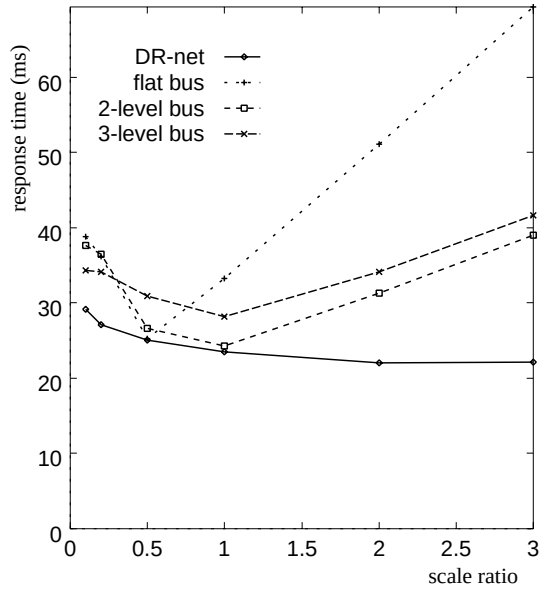

図 6.18: 書き込みのレスポンスタイム

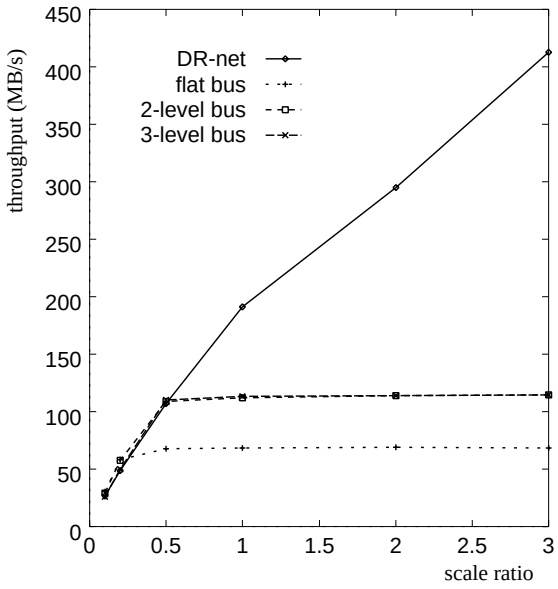

図 6.19: 書き込みのスループット

近い簡単なノードを組み合わせる方が望ましい場合が多いと思われる。

6.5 まとめ

本章では、システムの構成規模や通信性能に関するシミュレーション実験の結果から、DR-net 内部のネットワークや複数のインタフェースを用いる効果について検証した。

ディスクノード数の増加により、DR-net は高い性能を達成することができる。バスシステムでは性能が飽和してしまうことから、ディスクノード数に見合う性能を発揮できない。

インタフェースノードの増加に対しては、並行処理される要求数の増加にともないレスポンスタイムが増大するが、バスシステムと比べるとその差は小さい。スループットは、ディスクノード数の増加の場合と同様向上することが分かった。また、インタフェースノード数が小さいときには、読み出しのような比較的ディスクノードでの処理が軽い要求に対しては、要求の発行レートが上がることから、インタフェースノード周辺のリンクが通信のボトルネックとなり、性能が低下することが示された。従って、高性能なシステムの構成を考える際には、ディスクノード数と同様インタフェースノード数も重要な要素となる。

また、通信のバンド幅を変化させた実験では、読み出しのスループットの比較から、バスでディスクノードとインタフェースノードを結合する大規模な構成では大幅なバスの通信性能の向上が必要であり、ネットワークを用いることの有意性が示された。バスを用いる場合には、単純な階層構造ではなく fat-tree など何らかのネットワーク構成を採用することが自然と思われる。ディスクノード数とインタフェースノード数の比を一定に保った条件で規模を変化させた結果からは、DR-net が優れたスケーラビリティを持つことが示された。

これらの結果から、大規模構成を想定したディスクシステムにおけるネットワークの利用と複数のインタフェースを持つことの重要性が示され、DR-net が大規模な二次記憶システムの構成に適していることが示された。

第 7 章

書き込み性能の改善

第 5章ではパリティ分散によって RAID5 と同様の書き込み性能の向上が確認されたが、スループットモデルや実験結果から分かるように、パリティを分散保持する場合でも読み出しと比較すると書き込みの性能が低い。本章ではこの問題を取りあげ、DR-net の書き込み性能を改善するための方法について考察する。

7.1　書き込み処理の高速化

第 5章や第 6章の結果に見られるように、DR-net では読み出し性能と比べて書き込み性能が低い。DR-net では、1 つのディスクノードが FPG と SPG の 2 種類のパリティグループに属すため、1 つのデータブロックの更新には 6 回のディスクアクセス、2 つのノード間データ転送、3 回のパリティ計算が必要となる。ディスクアクセスは msec 単位の処理であり、性能が低い原因は、主に多数のディスクアクセスにあると思われる。一方、通信やパリティ計算は全体の処理に占める割合を考えれば比較的低コストの処理である。従って、DR-net の書き込み性能の改善を考える場合には、多少通信やパリティ計算の処理が増えても、ディスクアクセスのコストを低減することが重要である。

RAID 等の二次記憶システムの書き込み性能の改善に関してこれまで行なわれてきた主な研究としては、高速なアクセス速度を持つ記憶空間を併用することにより記憶階層を形成して全体を高速化するものや、ディスクアクセスのシークや回転待ち時間を削減するものがある。本章では、これらの 2 つのアプローチの例としてキャッシュメモリの利用とログベースの書き込みによる DR-net の書き込み性能の向上を考える。

7.2 ディスクキャッシュ

ディスクキャッシュは一部のディスクブロックをメモリ上に置き、それらに対する参照/更新をメモリ上で行なうことにより、ディスクアクセス回数を低減する。書き込み性能向上のためには write-back(書き戻し) キャッシュが用いられる。

RAID システムでは、キャッシュメモリはシステム全体を制御する単一のコントローラ(インタフェース) や論理的に単一とみなせるノードに置かれるのが一般的である。しかし DR-net では、キャッシュのための単一ノードを設置すると、キャッシュへのアクセスのための通信がその周辺リンクに集中し、ネットワークによる通信の分散という DR-net の利点が生かされない。従って、ネットワーク内のいくつかのノードに分散して設置することが望ましい。ここでは、キャッシュのための特別なノードを新たに付け加えることなく、インタフェースノードやディスクノードに設置することを考える。

ここで議論するインタフェースノードキャッシュおよびディスクノードキャッシュでは、基本的に参照/更新する全てのディスクブロックをキャッシュし、書き戻しはユーザからのアクセス要求処理とは非同期に行なわれると仮定する。また、ディスクキャッシュはある程度大きな容量が必要であるため、キャッシュメモリ自体は安価な揮発性メモリを仮定する。

キャッシュブロックのディスクへの書き戻しのタイミングや書き戻すキャッシュブロックの選択については、その最適性がディスクブロックのアクセスパタンに依存するため、ここでは触れない。

以後、キャッシュブロックの状態を次のように表現する。

free ディスクブロックを格納していない空のキャッシュブロック

clean キャッシュとディスクの内容が同一であり、書き戻しの必要がないキャッシュブロック

dirty キャッシュとディスクの内容が異なり、書き戻しの必要があるキャッシュブロック

7.2.1 インタフェースノード キャッシュ

インタフェースノードキャッシュは、ユーザから更新を要求されたデータブロックのみをキャッシュする。パリティの更新は書き戻し時に行なわれる。

キャッシュ内の dirty なデータはパリティによって保護されていないため、何らかの方法で電源故障などによるデータ喪失を防がなければならない。ここでは、別個の電源を持つ

2 つのキャッシュによりデータを 2 重化することで、データの喪失を防ぐ ことにする。

また、DR-net では、複数のインタフェースノード を持つことができるため、それらが持つ複数のキャッシュ間でキャッシュのコンシステンシを考慮する必要がある。この問題は 1 つのデータブロックは高々1 つのインタフェースノード でキャッシュされるようにすることで解決される。これにより、コンシステンシを保つと同時に、より多くの異なるディスクブロックをキャッシュすることが可能となり、キャッシュ容量を効率的に利用してヒット率を上げることができる。以下に、インタフェースノード キャッシュを用いた場合の具体的な動作について説明する。

書き込み動作

あるディスクブロックに対する書き込み要求が外部からインタフェースノード に到着すると、インタフェースノード はブロックが自分のキャッシュ内に存在するかどうか調べ、存在すればキャッシュにアクセスして処理を終る。

キャッシュに当該ブロックが存在しない場合のインタフェースノード の動作は以下のようになる。

1. そのブロックを格納しているディスクノード に対して、ブロックがどこかのインタフェースでキャッシュされているか問い合わせる。
2. キャッシュされていなければ、自分のキャッシュに書き込む。
3. どこかでキャッシュされていれば、キャッシュしているインタフェースにブロックを転送する。

ディスクノード の動作は以下のようになる。

1. あるブロックについてインタフェースノード からキャッシュの存在の問い合わせを受ける。
2. キャッシュしているインタフェースノード が存在すれば、その位置を返答する。
3. 存在しなければそのことを返答し、そのブロックのキャッシュ先として問い合わせ元のインタフェースノード を記憶する。

各ディスクノードは、自分が保持しているディスクブロックのうち、キャッシュされているものについてはそのキャッシュ位置を記憶している。これにより、キャッシュ位置管理のための処理が分散されると共に、1 つのブロックが複数のインタフェースでキャッシュされることが避けられる。

書き戻しとキャッシュエントリの削除

インタフェースノードからディスクへの書き戻しは、キャッシュを持たない通常の DR-net での書き込みと全く同じ動作により処理される。書き戻された dirty なブロックは clean なブロックとなり、キャッシュ内に留まる。clean なブロックは、free なキャッシュブロックが使い尽くされた後、新たなデータによって上書きされる。このとき、上書きされたブロックがキャッシュから削除されたことを当該ブロックを持つディスクノードに知らせる。ディスクノードはブロックのキャッシュが削除されたことを認識し、記憶していたブロックのキャッシュ位置情報を無効にする。

キャッシュエントリの更新とディスクノードのキャッシュ位置情報の更新のタイミングには、通信による遅延が生じる。このため、問い合わせで得られたキャッシュ位置にブロックを転送した時点で、既にそのキャッシュが削除されている可能性がある。そのようなキャッシュ位置の問い合わせに失敗した場合は、転送データを受け取ったインタフェースノードが再びキャッシュ位置の問い合わせを行ない、処理をやり直す。

7.2.2 ディスクノード キャッシュ

ディスクノードキャッシュは、個々のディスクノードが自分のディスクに保持されているディスクブロックをキャッシュする。

ユーザからのアクセス要求は、キャッシュにヒットした場合にディスクアクセスが必要なくなることを除き、キャッシュが存在しない場合と同様に処理される。ディスクノードキャッシュはアクセス処理に必要なブロックを全てキャッシュする。従って、インタフェースノードキャッシュと異なり、データブロックだけでなくパリティブロックもキャッシュされる。

キャッシュが存在しない場合と同様、個々の書き込み要求毎にパリティが更新されるため、各データブロックはパリティによってデータ喪失から保護されている。従って、インタフェースノードキャッシュの場合と異なり、キャッシュを 2 重化する必要はない。故障に

よりキャッシュの内容が失われた場合には、ディスク故障と同様に扱われ、データはパリティを用いて再構築される。

また、各ブロックのキャッシュ位置はそのブロックを持つディスクノード内に限定されるため、複数のキャッシュ間でコンシステンシを考慮する必要はない。

書き込み動作

キャッシュとディスクの関係は1つのディスクノードの中で閉じているため、ノード間にはキャッシュ制御のための特別な処理は必要ない。ユーザからのアクセス要求はインタフェース/ディスクノードの双方でキャッシュを持たない場合と全く同様に処理され、ディスクブロックへのアクセスの際に各ディスクノード内でキャッシュが用いられている点のみが異なる。

書き戻しとキャッシュエントリの削除

パリティ計算はキャッシュへの書き込みの時点で完了しているため、書き戻しに必要な処理は、データブロックおよびパリティブロック共にディスクへの書き込みのみである。書き戻しやキャッシュエントリの削除は、それぞれのディスクノードで独立に行なわれ、同期をとる必要はない。

7.2.3 書き込み要求の平均処理コスト

インタフェースノードキャッシュとディスクノードキャッシュについて、その1回の書き込みアクセス要求に対して必要な処理を、ノード間通信、ディスクアクセス、パリティ計算について見積もる。式中では、表7.1に示す表記を用いる。メモリの参照/更新など、以下に示されていない処理のコストは0と考える。また、パリティは全てのディスクに均等に分散されると仮定する。

インタフェースノードキャッシュを用いたときの平均コスト

1つの書き込みアクセスの平均コストは、次の3つの場合に分けて考えられる。ここでは、キャッシュ位置の問い合わせに失敗する確率は無視できるほど小さいとして、データの再転送のコストは考慮していない。

表 7.1: コストの表記に用いられるパラメータ

C_d : データブロック付ノード間通信の平均コスト
C : データブロックなしのノード間通信の平均コスト
D : 1 ブロックのディスクアクセスの平均コスト
X : ブロックを XOR するコスト
I : インタフェースノード数
n : 1 パリティグループ内のデータノード数
p : インタフェースノードキャッシュのヒット率
q : ディスクノードキャッシュのデータヒット率
r : ディスクノードキャッシュのパリティヒット率

- 要求が到着したインタフェースノードのキャッシュにヒット (確率: $1/I \cdot p$)

  この場合、直ちにキャッシュを更新して処理を終るため、コストは 0 である。

- 他のインタフェースノードのキャッシュにヒット (確率: $(I-1)/I \cdot p$)

  まずキャッシュ位置を問い合わせて、データを転送しなければならないから、

$$
\begin{aligned}
& (\text{キャッシュ位置問い合わせ}) + (\text{転送}) + (\text{acknowledgement}) \\
= \ & (C + C) + (C_d) + (C) \\
= \ & 3C + C_d
\end{aligned}
$$

- キャッシュミス (確率: $1-p$)

  キャッシュ位置の問い合わせと自ノードのキャッシュへの書き込みを行なう。新たなエントリをキャッシュに追加する場合、free あるいは clean なブロックが存在していることが必要である。free なブロックは初期状態にしか存在しないから定常状態では clean なブロックが必要となる。従って、dirty なブロックの書き戻しコストを考慮し、

1 回のキャッシュミスにつき、平均して 1 回の書き戻し動作が必要であるとする。

$$
\begin{aligned}
& \text{(問い合わせ)} + \text{(書き戻し)} + \text{(clean なキャッシュエントリの削除)} \\
= \ & \text{(問い合わせ)} + \text{(ノード予約} + \text{データノードでの処理} + \\
& 2(\text{パリティ転送} + \text{パリティノードでの処理})) + \text{(エントリ削除通知)} \\
= \ & 2C + (2C + C_d + 2D + X + C + 2(C_d + 2D + X + C)) + C \\
= \ & 6D + 8C + 3C_d + 3X
\end{aligned}
$$

これらを確率を考慮して合計すると、インタフェースノードキャッシュを用いた書き込み要求処理の平均コストは、次のようになる。

$$(6D + 8C + 3C_d + 3X) - (6D + \frac{5I+3}{I}C + \frac{2I+1}{I}C_d + 3X)p$$

ディスクノードキャッシュを用いたときの平均コスト

インタフェースノードキャッシュの実効容量は、2 重化しているために実容量の $1/2$ である。一方、ディスクノードキャッシュは 2 重化はしていないが、データブロックだけでなくパリティブロックも格納している。

n 個のデータノードと 1 つのパリティノードで構成されるパリティグループについてみると、パリティの容量はデータの $1/n$ であるが、どのデータが更新されてもパリティの更新が必要であるから、そのアクセス率は n 倍である。従って、キャッシュ中のデータとパリティの割合は、ほぼ 1:1 と考えられる。DR-net では、1 つのデータブロックに対し 2 つのパリティブロックが存在するから、データ:FPG パリティ: SPG パリティ=1:1:1 となり、データの実効容量は実容量の $1/3$ となる。これはインタフェースノードキャッシュの $1/2$ よりも小さいため、データブロックのヒット率は減少すると思われる。この減少率を m とすると、$q = mp$ となる。また [39] で、ある程度ヒット率が低いときにはヒット率は参照頻度に比例することが示されていることから、パリティはデータの n 倍参照されることを考慮して $r = nq = mnp$ となる。

1 つの書き込み処理の平均コストは、関係する 3 つのノード (データノード、FPG のパリティノード、SPG のパリティノード) でのキャッシュヒット/ミスの組み合わせで場合分けされる。以下に各場合の確率と処理コストを示す。D、P はそれぞれデータノード、パリティノードを表し、例えば「DP ヒット」はデータノードと 1 つのパリティノードでキャッシュがヒットする場合を表す。いずれの場合でも、変化するのはディスクアクセスのコストのみである。

- DPP ヒット (確率: $qr^2 = m^3n^2p^3$)

$$5C + 3C_d + 3X$$

- DP ヒット (確率: $2qr(1-r) = 2m^2np^2 - 2m^3n^2p^3$)

$$2D + 5C + 3C_d + 3X$$

- PP ヒット (確率: $(1-q)r^2 = m^2n^2p^2 - m^3n^2p^3$)

$$2D + 5C + 3C_d + 3X$$

- D ヒット (確率: $q(1-r)^2 = mp - 2m^2np^2 + m^3n^2p^3$)

$$4D + 5C + 3C_d + 3X$$

- P ヒット (確率: $2r(1-q)(1-r) = 2(mp - m^2n(n+1)p^2 + m^3n^2p^3)$)

$$4D + 5C + 3C_d + 3X$$

- ミス (確率: $(1-q)(1-r)^2 = 1 - m(2n+1)p + m^2n(n+2)p^2 - m^3n^2p^3$)

$$6D + 5C + 3C_d + 3X$$

これらを合計すると、書き込み要求処理の平均コストは

$$(6D + 5C + 3C_d + 3X) - 2m(2n+1)Dp$$

となる。

平均書き込み処理コストの比較

2 つのキャッシュ方式と、キャッシュを用いない場合の平均コストを表 7.2に示す。

この表から、最も重要と思われるディスクアクセスの低減について比較すると、インタフェースノードキャッシュとディスクノードキャッシュの低減数の比は、

$$ratio = \frac{\text{ディスクノードキャッシュの低減数}}{\text{インタフェースノードキャッシュの低減数}} = \frac{2m(2n+1)p}{6p} = \frac{m(2n+1)}{3}$$

表 7.2: 各キャッシュ方式の平均書き込み処理コストの比較

キャッシュなし	$6D + 5C + 3C_d + 3X$
インタフェースノード	$(6D + 8C + 3C_d + 3X) - (6D + \frac{5I+3}{I}C + \frac{2I+1}{I}C_d + 3X)p$
ディスクノード	$(6D + 5C + 3C_d + 3X) - 2m(2n + 1)Dp$

となる。十字型のパリティグループを用いる DR-net では $n = 4$ であるから、

$$ratio = 3m$$

また、m を単純に実効容量の減少率として計算すると、$m = \frac{1/3}{1/2} = 2/3$ から

$$rat\ i\ o = 2$$

となる。従って、これまでの議論の仮定が妥当といえる条件のもとでは、インタフェースノードキャッシュに比べてディスクノードキャッシュの方が 2 倍のディスクアクセス低減効果がある。

7.2.4 その他の性質の比較

平均書き込み処理コスト以外の性質を簡単にまとめておく。(IF)、(D) はそれぞれインタフェースノードキャッシュ、ディスクノードキャッシュに有利な点であることを示す。

(IF) 各キャッシュがフルアソシアティブ、あるいはディスクブロック番号をエントリのタグとして利用するセットアソシアティブキャッシュである場合、インタフェースノードキャッシュは各ディスクノード毎に割り当てるキャッシュ容量の比に制限がないため、アクセス先ノードの偏りに対してより柔軟に対応し、ヒット率を向上させることができる。それに対しディスクノードキャッシュでは、各ノードのキャッシュは自分のディスク内のブロックしか格納できないため、アクセスノードに偏りがある場合は実効容量が減少する。

(IF) インタフェースノードキャッシュは、2 重化によって故障時のデータ喪失を防ぐ。第 4章で示したように、2 重化は、パリティを分散して保持する DR-net の冗長化と比べて高い信頼性を提供する。従って、インタフェースノードキャッシュに格納されている

データは、ディスクノードキャッシュに格納されているデータと比べると、失われにくいと思われる。

(D) ディスクノードキャッシュは、キャッシュからディスクへの書き戻しを各ノード毎に独立に行なうことが可能であるため、ディスクアクセスに伴うノード間の同期が緩やかになる。インタフェースノードキャッシュでは、書き戻し処理で、キャッシュを用いない書き込みと同様の同期したディスクアクセスが必要となる。

(D) ディスクノードキャッシュは、キャッシュとディスクが同じノード内にあるため、書き戻し処理を行なう適切なタイミングの選択が考えられる。例えば、ディスクの利用率が低いときに書き戻し処理を行なうことが考えられる。インタフェースノードキャッシュでは、ディスクの使用状況に関する情報を持っていないため、負荷の高いディスクに対して書き戻し要求を発行してしまうことが考えられる。

(D) インタフェースノードキャッシュでは、キャッシュヒットした大半の場合にインタフェースノード間でのデータ転送が発生する。これは、インタフェースノード周辺のリンクに通信が集中する傾向をもたらし、その頻度が高ければ性能に悪影響を及ぼす可能性がある。ディスクノードキャッシュでは、そのような集中は発生しない。

(D) インタフェースノードキャッシュでは、各データブロックのキャッシュ位置に関する情報をディスクノードで保持しなければならない。アクセスの高速化のため、この情報はメモリ上に置かれることが望ましいが、そのメモリ量はキャッシュ容量に比例して増大する。ディスクノードキャッシュでは、余分なメモリは必要ない。

これらの諸条件を考慮すると、DR-net ではインタフェースノードキャッシュよりもディスクノードキャッシュを用いる方が、効果的に書き込みの高速化が実現できると思われる。

7.3 ログ

一般にキャッシュは、ヒット率が高くなければ効果が少ない。大規模なディスクシステムでは、システム全体の記憶容量も膨大で、キャッシュメモリはその一部を保持できるに過ぎない。従って、局所性が強いアクセスパタンについてのみ効果があると考えられる。

それに対し、参照の局所性それほど強くない場合でも書き込み性能を上げる方法として、Log-structured File System などに見られるような、ディスク上に散在するブロックに対す

る多くの小さな書き込みをバッファリングして、大きなシーケンシャルな書き込みに変換する方法がある。また、バッファリングしたデータを同じパリティグループに書き込むことで、パリティ更新を read-modify-write ではなく write のみで行なえるという利点もある。これらの利点から、ログを利用した手法は Log-structured File System(LFS)[28] をはじめとして広く用いられている。

本節では、ログによる書き込み性能向上の手法を DR-net に適用する際の問題点と適用した場合の効果を明らかにするため、ブロック単位の LFS 的アクセスについて考察する。

7.3.1 ログの利用の概要

ログを利用した書き込みでは、ディスクブロックの更新は、上書きではなくログへの追記という形で行なわれる。その際、ディスクへの書き込みを効率良く行なうため、ログは連続した大きなフリースペースであることが望ましい。そのためには、追記によって内容が古くなり、無効となったディスクブロックを集めて大きなフリースペースをつくり出す作業が必要である。

これら 2 つの動作について説明する [28]。

書き込み動作

ブロック単位の書き込みはバッファリングによりセグメント (segment) と呼ばれる単位にまとめられ、ディスクへの実際の書き込みはセグメント毎に行なわれる。これにより、複数の小さなディスクアクセスを大きな単一のアクセスに変換できる。

各ディスクブロックは、追記により最新の内容が記憶されている物理的な位置が変化するため、各ブロックの最新の位置を記憶する必要がある。データが更新されたときは、この位置情報も更新する。通常、これらの情報はメモリ上に置かれるため、参照/更新のコストは無視される [28]。

また、パリティグループ単位で書き込みを行なうことにより、パリティブロックは各データブロックについての read-modify-write ではなく、各データブロックの XOR 演算から生成されたブロックの単純な write アクセスで更新される。

図 7.1は、3 つのディスクからなるパリティグループで、2 つのデータディスクのブロック A1, B2 を更新する場合の処理を示している。ただし、簡単のため、セグメントは 1 つのディスクブロックからなるとしている。A1, B2 は上書きされず、使われていないフリース

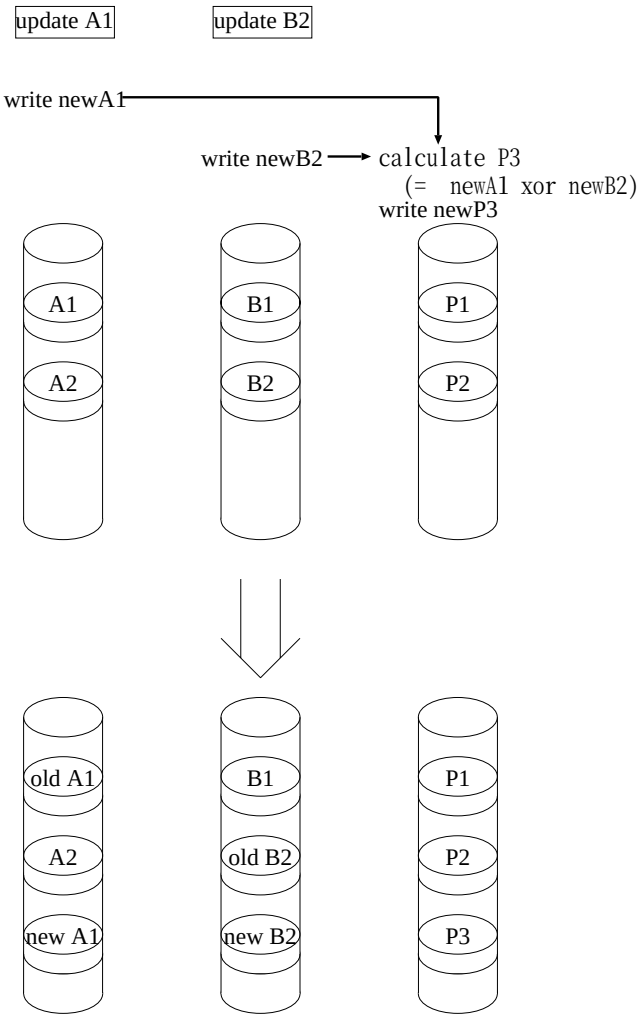

図 7.1: ログへの書き込み

ペースに新しい A1, B2 の値が書き込まれる。また、それらのパリティP3 が対応するパリティブロックに書き込まれる。A1, B2 の古い内容 old A1, old B2 は変更されずに残るため、P1, P2 のパリティブロックにも変更は必要ない。

セグメントクリーニング

あるデータブロックの新しい内容が追記されたとき、そのブロックの古い内容を記憶しているデータブロックはもはや必要ない。図 7.1の例では、old A1, old B2 がそのようなブロックである。無効なブロックと有効なブロックが混在するセグメントから、有効なブロックを

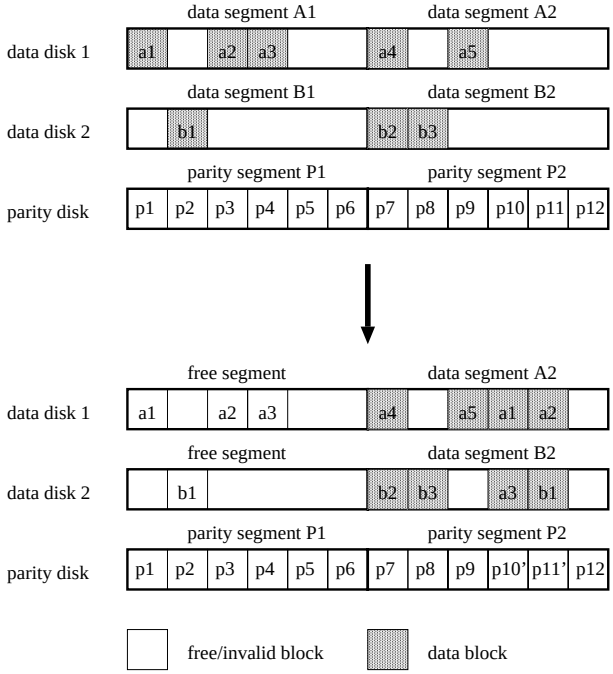

図 7.2: セグメントクリーニング

別のセグメントに移動し、セグメント全体を空にすることによって大きな連続するフリースペース (フリーセグメント) を確保できる。この作業をセグメントクリーニング (segment cleaning) と呼ぶ。

クリーニングを行なうためには、各セグメント内のブロックの使用状態の情報が必要である。この情報もメモリ上に置かれるため、参照/更新のコストは無視される。

図 7.2で、セグメント A1, B1 中の有効なデータブロックはセグメント A2, B2 にコピーされ、パリティブロック p10, p11 が更新される。フリーセグメントとなった A1, B1 の内容は変更されないため、パリティセグメント P1 の内容も更新の必要はない。

ログを利用した書き込みの特徴

ログを利用した書き込みの利点をまとめると、次の 2 点である。

- 複数のブロックアクセスをバッファリングすることにより、単一のセグメント単位の書き込みに変換できる。

- パリティグループ単位で書き込みを行なうことにより、ディスクから古いブロックの値を読むことなくパリティを生成できるため、パリティ演算およびディスクアクセスを削減できる。

反面、次のような欠点がある。

- 最新の内容を持つデータブロックの位置が変化するため、位置情報を管理しなければならない。
- セグメントクリーニングにより、フリーセグメントをつくり出す必要がある。また、各セグメント毎に使用ブロック数などを管理する必要がある。

7.3.2 DR-net での問題点

DR-net にログを利用した書き込みの高速化を適用する場合、FPG と SPG の 2 つのパリティグループが重なりあっていることが問題となる。

1 つのブロックが 1 つのパリティグループに属す RAID では、パリティグループ単位での書き込みを行なうことにより、新しいデータのみからパリティを生成できる。ところが、2 種類のパリティグループが存在する DR-net では、1 つのデータブロックに対して 2 つのパリティブロックを更新しなければならない。図 7.3は、FPG にパリティグループ単位で書き込んだときの、2 つのパリティブロックの生成を示している。パリティブロック P1 は、データブロック A, B, C, D の XOR 演算で生成できるが、パリティブロック P2 は read-modify-write によって更新する必要がある。従って、古い D の内容を読む必要がある。図では、D の属す SPG しか示していないが、A, B, C の属す SPG のパリティブロックもそれぞれ更新しなければならないから、結局パリティグループ単位で書き込む場合にも全てのデータブロックで古い値の読み出しが必要となる。

そのため、RAID の場合と異なり、書き込むデータセグメントやパリティセグメントをあらかじめ読み出しておく必要がある。

このように、DR-net におけるパリティグループ毎の書き込みでは、ディスクアクセス数がほとんど減少せず、さらにインタフェースノード内にバッファとパリティ生成のための XOR 演算器が必要となることを考慮すると、そのメリットは少ないと思われる。そこで、以後はブロック単位のアクセスのみを考える。

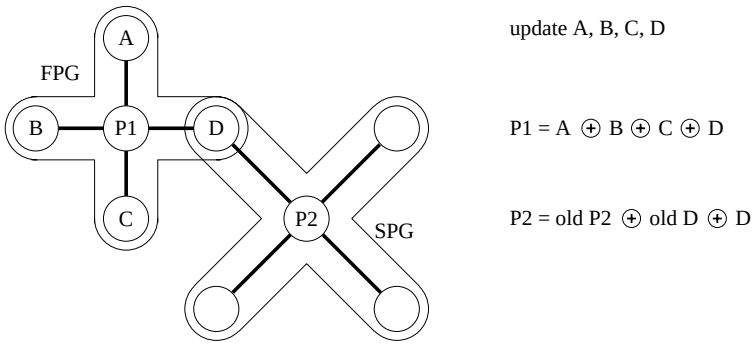

図 7.3: ログを用いた場合の 2 つのパリティ更新

7.3.3 アクセスコストの見積もり

書き込みのコスト

1 つのセグメントに対する書き込みの動作は次のようになる。

1. データノードが書き込むセグメントを選択し、選択したセグメントをパリティノードに通知すると共に、ディスクからバッファに読み出す。
2. パリティノードは通知を受け取ると、ディスクからそのデータセグメントに対応するパリティセグメントをバッファに読み出す。
3. データノードは、インタフェースノードから書き込み要求を受け取る。
4. セグメント内のブロックとの差分パリティを計算し、パリティノードに送る。その後、セグメントのブロックを更新する。
5. パリティノードは、差分パリティとセグメント内のブロックからパリティを計算し、セグメントに書き込む。書き込みが終了したら返答する。
6. データノードは、書き込み処理が終ったことをインタフェースノードに通知する。
7. 3–6 をセグメントが満たされるまで繰り返す。
8. データノード、パリティノードでそれぞれセグメントをバッファからディスクに書き込む。

1 セグメントが平均 b ブロックで満たされるとすると、1 つのデータセグメントへの書き込みのコスト W は、

$$\begin{aligned} W &= 2C + 3D_L + (2C + C_d + X + 2(C_d + X + C) + C)b + 3D_L \\ &= 6D_L + (5b + 2)C + 3bC_d + 3bX \end{aligned}$$

となる。ただし、D_L はセグメント単位のディスクアクセスを表す。それ以外は表 7.1の表記に従う。

セグメントクリーニングのコスト

1 つのセグメントをクリーニングするときの動作は次のようになる。

1. データノードは、クリーニングするセグメントを選択する。
2. セグメント内の有効なデータブロックを読み出し、それに対して上記の書き込み動作を行なう。
3. 有効な全てのデータブロックに対して、2 を繰り返す。

クリーニングする 1 セグメント内に平均で v 個の有効なデータブロックがあるとすると、セグメントクリーニングのコスト S は

$$S = v(D + 2C + 2C_d + 3X)$$

となる。

1 回あたりの書き込み要求の処理コスト

ログを用いる場合は、セグメントクリーニングのコストも含めて考える必要がある。1 つのセグメントが新しいデータで満たされると、1 つのフリーセグメントが必要になるから、セグメントが満たされる毎に平均して 1 回のクリーニングが行なわれるとする。このとき、2 つのセグメントクリーニングの間で処理される書き込み要求数は、1 セグメントのブロック数を s とすると、$b = s - v$ となる。1 回当たりの平均書き込みコスト w_{avg} は

$$\begin{aligned} w_{avg} &= \frac{W + S}{b} \\ &= \frac{6D_L + vD + (5s - 3v + 2)C + (3s - v)C_d + 3sX}{s - v} \end{aligned}$$

となる。一方、ログを用いない DR-net の書き込みコストは表 7.2から、$6D+5C+3C_d+3X$ である。ログを用いる場合、ディスクアクセス以外のコストは若干増加している。

ディスクアクセスについては、ログを用いない場合は $6D$、ログを用いると $\frac{6D_L+vD}{s-v}$ である。D_L は D の s 倍のデータを転送するが、s 回の D と比較するとシーク時間や回転待ち時間が減少するため $D_L < s$ Dである。$D_L = xD, 1 < x < s$ とすると、ログを用いたときのディスクアクセスコストは

$$\frac{6D_L + vD}{s-v} = \frac{6x+v}{s-v}D$$

となる。従って、$\frac{6x+v}{s-v} < 6$、すなわち

$$v < 6(s-x)/7$$

であればログを用いる効果がある。1 回のディスクアクセスの平均のシーク時間と回転待ち時間の合計を i、1 ブロックの転送時間を t とすると、$x = \frac{i+st}{i+t}$ であるから、次式が成り立てば、ログを用いる効果がある。

$$v < \frac{6(s-x)}{7} = \frac{6(s-1)i}{7(i+t)} \tag{7.1}$$

また、式 (7.1) は

$$\frac{v}{s-1} < \frac{6}{7} \times \frac{i}{i+t}$$

となるので、クリーニングするセグメントの有効ブロックの割合が $6/7 \simeq 85\%$ を越える場合には、転送レートが極めて大きい場合でもクリーニングのオーバヘッドが大きくなり、ディスクの転送速度に関わらずログを利用する効果がないことが分かる。ただし、ここではメモリ転送やセグメントの管理コストを考慮していないため、これらのコストがディスクアクセスコストに比較して無視できない場合には、もっと低い割合でもログを用いる効果はなくなる。

また、v が大きい場合にはクリーニングするセグメント全体を読み込み、vD を D_L に変換することも考えられる。この場合、1 回のアクセス要求当たりのディスクアクセスコストは $\frac{7D_L}{s-v}$ となる。これを $\frac{6D_L+vD}{s-v}$ と比較すると、

$$\frac{6D_L + vD}{s-v} - \frac{7D_L}{s-v} = \frac{v-x}{s-v}D$$

従って、$v > x$ のときにセグメント全体を読み込む効果がある。

7.4 関連研究

ディスクキャッシュの利用に関するいくつかの研究がある。[13] では、キャッシュの位置や置換アルゴリズムなどについて議論し、単一コントローラにキャッシュをおくシステムでさまざまなアクセスパタンを用いてアルゴリズムの評価を行なっている。しかし、冗長情報の管理については考慮されていない。また、[17][16] では RAID5 を対象としてデステージアルゴリズムや故障モードでの性能などについて考察している。システムのコントローラは 2 重化あるいは不揮発性キャッシュを持っており、本章で述べたインタフェースノードキャッシュに近いが、複数のキャッシュが存在するときのコンシステンシについては言及されていない。

また、キャッシュメモリ以外による階層化の例としては [15] や、Hot mirroring[20], HP AutoRAID[33] で、RAID レベル 5 と RAID レベル 1 の組み合わせによる手法などが研究されている。RAID レベル 5 とレベル 1 の組み合わせによる手法は、記憶効率が良く書き込みコストが大きい構成の上に、記憶効率が悪く書き込みコストが小さい構成をキャッシュ的に用いる階層的な構成方式であり、DR-net でも下層の構成が RAID レベル 5 でないことを除けば、そのまま適用できる手法である。

Parity Logging[30] では、書き込み処理の際にパリティブロックを直ちに更新せず差分パリティを連続領域にログとして残し、まとまった量のパリティを一度に更新する。これにより、パリティ更新に伴うディスクのシークや回転待ち時間を削減する。また、Floating Parity ではパリティブロックを上書きせず、格納する位置を変化させることでディスクの回転待ち時間を削減する。Hot mirroring[20], HP AutoRAID[33], Zebra[11] など多くのシステムで採用されている Log-structured File System(LFS)[28] の考え方は、データもパリティもログに記録することでディスクアクセスを効率良く行なう。この手法の DR-net への適用については、本章で議論した。また、Virtual Striping[21] では、セグメントクリーニングを copying garbage collection ではなく、ストライプの動的再編成によって行なうことにより LFS をさらに高速化している。

7.5 まとめ

DR-net の問題点である書き込み性能の向上について、ディスクアクセスのコストの低減に着目し、ディスクキャッシュの利用とログの利用について考察した。

キャッシュはインタフェースノードに置く方法と、ディスクノードに置く方法の 2 つについて検討した。インタフェースにキャッシュを置く場合には、DR-net では複数のインタフェースノードがあることからキャッシュコンシステンシを考慮しなければならない点に言及し、具体的な動作を説明した。また、両方式での書き込み処理の平均コストを示し、ディスクノードキャッシュを用いる方がディスクアクセスコストを減らせることを示した。また、ディスクアクセスコスト以外のいくつかの点でも、ディスクノードキャッシュが優れていることを示した。

また、ログを用いる書き込みに関しては、DR-net では 2 つのパリティグループが重なりあっていることから、RAID と異なりパリティ生成の簡略化によるディスクアクセス数の低減ができないことを示した。また、書き込みおよびセグメントクリーニングのコストにもとづいて、1 回の書き込み要求当たりの平均コストを、セグメントサイズ、シーク/回転待ち時間、転送レートから具体的に見積もり、書き込み性能が改善されるための条件を示した。

式 (7.1) から、シークや回転待ちの時間に対してデータ転送時間が短いディスクを用い、1 セグメントのブロック数を増やすことにより、v の上限を上げることができる。将来、ディスクのシークや回転待ち時間が現在よりも大幅に減少することは考えにくいが、転送レートは記録の高密度化により向上すると思われる。セグメントサイズに関してはバッファ容量を考慮しなければならないが、メモリ価格の下落傾向が続くとすればセグメントサイズの拡大はコスト的にも可能となる。従って、ログを用いて DR-net の書き込み性能を向上させる手法は、今後より有効となると考えられる。

第 8 章

結論

本研究では並列ディスクシステム、Data Reconstruction networks(DR-net) について、性能および信頼性の両面から検証し、またその問題点の指摘と解決策の提案を行なった。主に次の 4 つの点に関して検証、考察した。

(1) 信頼性

(2) 動作方式の性能への影響

(3) 構成規模や通信性能の変化とシステム性能の関係

(4) 書き込み性能の向上

(1) では、冗長率をそろえた 25 台のディスクノードを持つ構成で具体的に MTTF を比較した結果、従来から提案されているパリティグループを用いた DR-net は、いずれも RAID レベル 3 〜 5 よりも高い信頼性を有することが明らかになった。特にパリティを分散させなかった場合には、DR-net の冗長情報利用により RAID レベル 6 やレベル 1 をしのぐ高信頼なシステムが実現されることを示した。また、2 つのパリティ分散方式については、MPN を用いた方が信頼性は高いが、MPG と MPN の間には信頼性に関して大きな差はないことが示された。

一般的な規模での信頼性に関しては、MTTF による比較は困難であるが、十字型のパリティグループを用いる構成についてマスク可能な最大故障数がほぼ規模に比例して増加することが分かった。パリティを固定配置した場合には、RAID レベル 6 よりもマスク数は大きくなり、MPN で分散した場合でもほぼ RAID レベル 6 と同数の故障がマスクできるこ

とが示された。各構成方式のカバリッジの大小関係が規模の変化にともなって大きく変化しないとすれば、パリティを固定した DR-net は一般的に RAID レベル 6 よりも高い信頼性を持ち、パリティを分散した場合でも RAID レベル 3 〜 5 よりも十分に高い信頼性を持つと考えられる。

信頼性と記憶効率の関係については、DR-net でも記憶効率の柔軟な選択が可能であることが可能であることを示し、いくつかの構成例を挙げた。それらの信頼性を検討した結果、ほぼ同数のディスク台数で構成される RAID レベル 1 や同じ冗長率を持つ RAID レベル 6 よりも高い信頼性の維持が可能であることを示した。また、それらの構成においてパリティを分散した場合でも、ディスク台数の違いを考慮すれば RAID レベル 6 と同程度の信頼性が得られると思われ、冗長率を低減した場合でも RAID レベル 3 〜 5 をしのぎ RAID レベル 6 に匹敵する高信頼な DR-net の構成が可能であることが示された。

動作方式間の特性について、(1) では信頼性の違いを示したが、(2) ではさまざまなパリティ分散方式やデータ再構築戦略が DR-net の性能に与える影響ついて、実験機を用いた実験結果から評価した。トランスピュータと小型ディスクを用いた DR-net を実装にあたり、データとパリティの不整合についてその解決法を考察した。RAID システム等で採用されているロックによる排他制御は処理コストの増大から DR-net では適切ではないことを示し、付加コストの少ない手法ディスクブロックのバージョン管理による楽観的な手法を採用した。

性能評価では、故障が存在しない場合のシステム内部のスループットのモデルを示し、実験結果からは、DR-net では 1 つのデータ更新に対して 2 つのパリティ更新が伴うため書き込み性能に関してはパリティを分散配置することが重要であることが確認された。

MPG と MPN の比較では、故障が存在しないときの書き込みではパリティグループ内のノード間距離の短い MPG の方がより効果的であることを示した。しかし、故障が存在するときは MPG でのデータ再構築には平均して MPN よりも多くのディスクアクセスが必要であり、MPN の方が良い性能を示す傾向があることがわかった。2 つのデータ再構築戦略に関しては、再構築の際に比較的少ないノードを必要とする LRS がスループット、レスポンスタイムの両面で有利であることを示した。

DR-net の性能に関して、(2) では特定の規模について評価実験を行なったが、(3) ではノード数や通信バンド幅を変化させた場合の性能への影響について検証した。、ディスクノード数やインタフェースノード数が増加したとき、バスシステムでは性能が飽和してしまう

のに対し、DR-net では増加した台数に見合う性能向上が実現されることを示した。インタフェースノードの増加に対してはレスポンスタイムが増大するが、バスシステムと比べるとその増加は少ない。また、インタフェースノード数が小さいときには、インタフェースノード周辺のリンクが通信のボトルネックとなり性能が低下する場合があるため、ディスクノード数と同様インタフェースノード数も重要な要素となることを示した。さらに、通信のバンド幅を変化させた読み出し性能に関する結果からは、特にスループットに関して、DR-net と同程度の性能をバスシステムで実現する場合にはバスの通信性能の大幅な向上が必要であることが明らかとなりった。バスでディスクノードとインタフェースノードを結合する構造は大規模な構成には適せず、内部ネットワークの利用が有効であることが示された。

また、いくつかの評価結果では性能がパラメータの変化にともなって単調には増減しない結果が見られ、システムの構成条件の変化にともなって要求を処理する時間に対して支配的となる要因も変化することが示された。

(4) では、DR-net でのディスクキャッシュやログの利用について考察した。キャッシュはインタフェースノードに置く方法とディスクノードに置く 2 つの方法を検討し、それぞれの平均書き込みコストを示した。その結果、ディスクノードキャッシュを用いる方がディスクアクセスコストを減らせることが明らかとなった。また、ノード間の同期の緩和や書き戻しタイミングの選択などの点でも、ディスクノードキャッシュが優れていることを示した。

ログを用いる書き込みに関しては、書き込みおよびセグメントクリーニングのコストを見積もり、1 回の書き込み要求当たりの平均コストからログの利用が有効であるための条件を示した。その結果、DR-net では各パリティグループが重なりあっているためにパリティ生成の簡略化によるディスクアクセスの低減ができないにもかかわらず、ログを用いる高速化は DR-net でも十分に適用可能であることを示した。

以上の結果から、DR-net によって、RAID レベル 3 〜 5 よりも高い信頼性を持ち、またバスを用いる RAID レベル 6 よりも高い性能を達成する大規模な二次記憶システムを構築できることが示された。これにより計算機システム全体の高速化がもたらされ、また大量のデータを扱う新たなアプリケーションやシステムの開発が期待される。

しかし、DR-net の研究に関する今後の課題として、次のようなものが残されている。

(1) では、厳密に信頼性の優劣を評価するためには、一般的な規模で MTTF を評価する必要がある。その場合、カバリッジの厳密値の計算は困難であることから、近似値の利用や、各故障数におけるカバリッジについて DR-net/RAID 間の大小関係のみを示すことが考えられる。また、故障ディスクの MTTR を用いたシステムの MTBF による評価も課題である。

(2), (3) では、パリティグループやネットワークトポロジの変化に伴う性能への影響を調査する必要がある。これまで、ネットワークは 2 次元トーラスに限って議論してきたが、その理由はネットワーク内で局所的に配置できるコンパクトな FPG, SPG が知られていたためである。他のネットワークにおいても適切なパリティグループの配置が得られれば、ネットワーク直径などの点で 2 次元トーラスより優れたトポロジを利用することにより、性能が向上する可能性がある。また、データ再構築戦略に関しては、システム負荷の状況に応じて LRS と ERS を動的に切替えることが考えられる。また、同じ LRS でも FPG と SPG のどちらを優先的に利用するかについて、動的に選択することにより再構築負荷を減らすことが考えられる。このような動作方式についても研究が望まれる。

(4) では、実際にキャッシュやログを用いたときの性能の向上を検証する必要がある。その際、参照の局所性と性能の関連についても解析が必要である。

また、故障ディスクの代わりに使用する予備ディスクの利用や、リンクなどのディスク以外の故障への対応についても課題として残されている。これらの課題を解決することにより、DR-net の高性能、高信頼性がより明らかになり、優れた二次記憶システムとして実現されると信ずる。

謝辞

本研究を進めるに当たり、終始御指導を賜わり、また幾多の発表の機会を与えて頂きました横田治夫助教授に心から感謝致します。また、適切な御示唆、御指導を賜わりました日比野靖教授に深く感謝致します。

堀口進教授、中島達夫助教授、ならびに東京大学生産技術研究所の喜連川優助教授には本論文をまとめるに当たり御指導を頂きました。

また、日頃から有益な御助言を頂き、多面に渡って励まして頂いた杉野栄二博士、宮崎純博士、ならびに丹康雄博士に感謝致します。

最後に、貴重な御意見、御討論を頂きました日比野・横田研究室の皆様をはじめとする多くの方々の御援助に対し、厚く御礼申し上げます。

参考文献

[1] Mario Blaum, Jim Brady, Jehoshau Bruck, and Jai Menon. EVENODD: An Optimal Scheme for Tolerating Double Disk Failures in RAID Architectures. In *Proc. of the 21st ISCA*, pp. 245–254, 1994.

[2] Mario Blaum, Jim Brady, Jehoshua Bruck, and Jai Menon. EVENODD: An Efficient Scheme for Tolerating Double Disk Failures in RAID Architectures. *IEEE Transactions on Computers*, Vol. 44, No. 2, pp. 192 – 202, Feb 1995.

[3] Walter A. Burkhard and Jai Menon. Disk Array Storage System Reliablity. In *Digest of Paper FTCS 23*, pp. 432 – 441, 1993.

[4] Pei Cao, Sween Boon Lim, Shivakumar Venkataraman, and John Wilkes. The TickerTAIP parallel RAID architecture. In *Proc. of the 20th ISCA*, pp. 52 – 63, 1993.

[5] Peter M. Chen et al. RAID: High-Performance, Reliable Secondary Storage. *ACM Computing Surveys*, Vol. 26, No. 2, pp. 145 – 185, Jun 1994.

[6] Ann L. Drapeau, Ken W. Shirriff, and John H. Hartmann. RAID-II: A High-Bandwidth Network File Server. In *Proc. of the 21st ISCA*, pp. 234–244, 1994.

[7] Thomas E.Anderson, David E. Culler, David A. Patterson, and the NOW team. A Case for NOW(Networks of Workstations). In *IEEE Micro*, pp. 54–64, Feb 1995.

[8] Garth A. Gibson, David F. Nagle, Khalil Amiri, Fay W. Chang, Eugene M. Feinberg, Howard Gobioff, Chen Lee, Berend Ozceri, Erik Riedel, David Rochberg, and Jim Zelenka. File server scaling with network-attached secure disks. In *Proc. of the ACM Int'l Conf. on Measurement and Modeling of Computer Systems*, 1997.

[9] Garth A. Gibson, Daniel Stodolsky, Fay W. Chang, William V. Courtrigght II, Chris G. Demetriou, Eka Ginting, Mark Holland, Qingming Ma, LeAnn Neal, R. Hugo Patterson, Jiawen Su, Rachad Youssef, and Jim Zelenka. The scotch parallel storage systems. In *Proc. of the IEEE CompCon Conf.*, 1995.

[10] Garth Gibson, Lisa Hellerstein, Richard M Karp, and David A. Patterson Randy H. Katz. Coding Techniques for Handling Failures in Large Disk Arrays. In *Proc. of the 3rd Intn'l Conf. on ASPLOS*, 1989.

[11] John H. Hartman and John K. Ousterhout. The Zebra Striped Network File System. In *Proc. of the Symp. on Operating System Principles*, 1993.

[12] William V. Courtright II, Garth Gibson, Mark Holland, and Jim Zelenka. A structured approach to redundant disk array implementation. In *Proc. of the Int'l Computer Performance and Dependability Symp.*, Sept 1996.

[13] Ramakrishna Karedla, J. Spencer Love, and Bradley G. Wherry. Caching strategies to improve disk system performance. *IEEE Computer*, Vol. 27, No. 3, pp. 38–46, Mar 1994.

[14] Edward K. Lee and Randy H. Katz. The Performance of Parity Placements in Disk Arrays. *IEEE Transactions on Computers*, Vol. 42, No. 6, pp. 651 – 664, Jun 1993.

[15] Chung-Sheng Li, Ming-Syan Chen, Philip S. Yu, and Hui-I Hsiao. Combining Replication and Parity Approaches for Fault-Tolerant Disk Arrays. *IEEE Parallel and Distributed Processing*, 1994.

[16] Jai Menon. Performance of RAID5 Disk Arrays with Read and Write Caching. *Distributed and Parallel Database*, Vol. 2, pp. 261–293, 1994.

[17] Jai Menon and Jim Cortney. The Architecture of a Fault-Tolerant Cached RAID Controller. In *Proc. of the 20th ISCA*, pp. 76 – 86, 1993.

[18] Jai Menon and Dick Mattson. Comparison of Sparing Alternatives for Disk Arrays. In *Proc. of the 19th ISCA*, pp. 318 – 329, 1992.

[19] Arif Merchant and Philip S. Yu. Design and Modeling of Clusterd RAID. In *Digest of Paper FTCS 22*, pp. 140 – 149, 1992.

[20] Kazuhiko Mogi and Masaru Kitsuregawa. Hot mirroring: A method of hiding parity update penalty and degradation during rebuilds for RAID5. In *Proc. of ACM SIGMOD Conference*, pp. 183–194, Jun 1996.

[21] Kazuhiko Mogi and Masaru Kitsuregawa. Virtual striping: A storage management scheme with dynamic striping. *IEICE Transactions on Information and Systems*, Vol. E79-D, No. 8, pp. 1086–1092, Aug 1996.

[22] Richard R. Munts and John C. S. Lui. Performance Analysis of Disk Arrays Under Failure. In *Proc. of the 16th VLDB*, pp. 162 – 173, 1990.

[23] John K. Ousterhout, Andrew R. Cherenson, Frederick Douglis, Michael N. Nelson, and Brent B. Welch. The sprite network operating system. *IEEE Transactions on Computers*, Vol. 21, No. 2, pp. 23–36, February 1988.

[24] Chan-Ik Park. Efficient placement of parity and data to tolerate two disk failures in disk array systems. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 6, No. 11, pp. 1177–1184, Nov 1995.

[25] Chan-Ik Park. Striping in a disk array with data/parity placement scheme rm2 tolerating double disk failures. *IEICE Transactions on Information and Systems*, Vol. E79-D, No. 8, pp. 1072–1085, Aug 1996.

[26] David A Patterson, Garth Gibson, and Randy H. Katz. A Case for Redundant Arrays of Inexpensive Disks(RAID). In *Proc. of ACM SIGMOD Conference*, pp. 109–116, Jun 1988.

[27] Erik Riedel and Garth Gibson. Understanding customer dissatisfaction with underutilized distributed file servers. In *Proc. of the 5th NASA Goddard Space Flight Center Conf. on Mass Storage Systems and Technologies*, 1996.

[28] Mendel Rosenblum and John K. Ousterhout. The Design and Implementation of a Log-Structured File System. In *Proc. of the 13th ACM Symp. on Operating Systems Principles*, pp. 1–15, Oct 1991.

[29] Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *IEEE Computer*, Vol. 27, No. 3, pp. 17–28, Mar 1994.

[30] Daniel Stodolsky, Garth Gibson, and Mark Holland. Parity Logging Overcoming the Small Write Problem in Redundant Disk Arrays. In *Proc. of the 20th ISCA*, pp. 64 – 75, 1993.

[31] Seishi Tomonaga and Haruo Yokota. An Implementation of a Highly Reliable Parallel-Disk System using Transputers. In *Proc. of the 6th Transputer/Occam Intn'l Conf.*, pp. 241–254. IOS Press, Jun 1994.

[32] John Wilkes. The DataMesh research project. In P. Welch et al., editor, *Transputing '91*, pp. 547 – 553. IOS Press, 1991.

[33] John Wilkes, Richard Golding, Carl Staelin, and Tim Sullivan. The HP AutoRAID hierarchical storage system. In *Proc. of 15th ACM Symp.on Operating Systems Principles*, pp. 96–108, Dec 1995.

[34] Haruo Yokota. DR-nets: Data-Reconstruction Networks for Highly Reliable Parallel-Disk Systems. In *Proc. of 2nd Workshop on I/O in Parallel Computer Systems*, pp. 105 – 116, Apr 1994. (Also in ACM Computer Architecture News Vol.22, No.4 Sep. 1994).

[35] Hauro Yokota and Seishi Tomonaga. The Performance of a Highly Reliable Parallel Disk System. In A. De Gloria, M. R. Jane, and D. Marini, editors, *Proc. of the World Transputer Congress '94*, pp. 147–160. The Transputer Consortium, IOS Press, Sep 1994.

[36] 横田治夫. RAID のネットワーク上への展開と信頼性向上. 信学技法 CPSY 93–11, 電子情報通信学会, Apr 1993.

[37] 横田治夫. データ再構築ネット (DR-net) における不均衡対策. 信学技法 FTS 93–20, 電子情報通信学会, Aug 1993.

[38] 横田治夫, 友永誠史. 高信頼並列ディスクプロトタイプのアクセス性能. 信学技報 FTS 94–27, 電子情報通信学会, Jul 1994.

[39] 小島信. 冗長ディスクアレイ用耐故障性分散ディスクキャッシュに関する研究. Master's thesis, 北陸先端科学技術大学院大学, 1996.

[40] 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関する研究. Master's thesis, 北陸先端科学技術大学院大学, 1994.

本研究に関する発表論文

[1] 味松 康行, 横田 治夫, “並列ディスクシステムのパリティグループの構成の変化と信頼性の比較”, 電子情報通信学会技術報告, FTS95-34, (1995.8).

[2] 味松 康行, 横田 治夫: “ネットワーク結合並列ディスクの外部インタフェース”, 電子情報通信学会技術報告, CPSY95-80, DE95-66, (1995.12).

[3] 味松 康行, 横田 治夫: “相互結合網構成の並列情報サーバ上のファイル管理”, 情報処理学会研究報告, DBS106-3, (1996.1).

[4] 味松 康行, 横田 治夫: “ネットワーク結合並列ディスクにおける耐故障制御の影響”, 情報処理学会論文誌, Vol.37, No.7, pp.1419-1428, (1996.7).

[5] 味松 康行, 横田 治夫: “耐故障並列ディスクシステムにおける通信衝突の影響”, 電子情報通信学会技術報告, FTS97-31, (1997.8).

[6] 味松 康行, 横田 治夫: “大規模並列ディスクの結合形態と性能の評価”, 情報処理学会第 55 回全国大会論文集, 6F-2, (1997.9).