静　岡　大　学　　博　士　論　文

ディスクシステム分散型アーキテクチャと そのビデオオンデマンドへの応用に関する研究

１９９８年２月

大学院電子科学研究科
電子応用工学専攻

中村　俊一郎

論文の要旨

電子計算機の高性能化は、近年、マイクロプロセッサ技術を中心に急速な進展を遂げている。Mooreの法則に代表されるLSIチップの集積度向上が進み、これによりスーパースカラ機構の実装が可能となり、これと並列化コンパイラと合わせて実現される、命令行規模の並列処理技術等の進展は著しい。一方マルチプロセッサによる並列処理も、従来汎用計算機の最上位機の位置付けで商用化されてきたが、今日UNIXサーバーやさらに安価なパソコンサーバーに主役が受け継がれ多くの商用機が出始めている。これらは主記憶とディスクを共有する全共有方式のマルチプロセッサ上で、複数のタスクないしスレッドを1個づつ個々のプロセッサが並列に処理する方式が中心である。ないしは主記憶を共有しないクラスター接続型もあるが、これはプロセッサの1つがダウンした場合に他のプロセッサが代行するという、高信頼性のみを追求するものであるため、ここでは対象外とする。さて、本来1つのタスク系列となるべきシーケンシャルな処理を均等に横方向に複数に分割して、複数のプロセッサで並列実行する技術は、ビデオ配信とかデータベース処理といった応用毎に研究されているのが実状であり、まだ多くの研究課題が残されている。この手段としては一般に、主記憶とディスクを全く共有しない無共有方式のマルチプロセッサが使われる。本論文は、ビデオサーバー応用とデータベース応用のそれぞれの処理の高速化を目的として、それぞれに対してこのような無共有型マルチプロセッサの新しい方式を提案し、その試作結果について考察する。

第1章では従来の研究を概観し、本研究の特徴を述べる。

第2章では、無共有方式マルチプロセッサで、プロセッサ間を高速バス接続することを特徴としたデータベースマシンを提案し、その試作結果について述べる。ハードウェア構成、ソフトウェア構成の特徴、その上でのジョイン、プロジェクション等のリレーショナルデータベース（RDB）処理の並列実行手法、RDBに特化して高速化したディスク及びディスクキャッシュアクセス法、そして障害回復法について述べる。次いで試作機を拡張ウィスコンシンベンチマーク等により性能評価した結果について述べる。ここで価格的に二桁上の他のマシンと同等の性能であること、又商用版の性能評価結果では、価格的に同規模の他のマシンに比べて約40倍高速であること等が示される。このマシンの適用事例の一

つとして、その高速性により、インデックスを使わないのみならず、無条件中間一致検索を使う、情報検索システムの実例を示し、①ユーザーにとって使いやすい、②データベースの設計、維持管理が省力化される、といった利点があることを示す。

第3章ではLAN上でビデオデータ転送の評価実験を行い、市販サーバーでビデオサーバーを構成した場合にビデオ配信性能上、CPU、バス、ディスクのどこがボトルネックとなるか調査する。この結果、現状の市販コンポーネントで構成する限りにおいては、サーバーのCPUがボトルネックになるという新事実を発見する。一方ディスク性能の観点からは、ビデオサーバーにはRAIDディスクアレイが適しているという筆者の持論を、3つの利点を示し分析を加える。以上を合成するものとして、RAIDの、ディスクをサーバーに置き換えて無共有方式マルチプロセッサ構成とした、分散RAID0型ビデオサーバーを提案する。この試作にあたり、ハード、ソフト共に出来る限り市販品を流用して実現する必要があり、そのためのキーポイントとなる、TSR機能を用いたソフトウェアの実現手法を説明する。試作機の評価の結果、当初の期待通りサーバー台数にほぼ比例したビデオ配信性能を実現していること、又画質も非常に良好であることを確認する。しかも約43ビデオストリームという現実に近い負荷状況まで試験して、その現実性をアピールする。

第4章では性能に加えて高信頼性も追求した分散RAID4及び分散RAID5型ビデオサーバーと呼ぶ方式を提案し、それぞれ試作ないしシミュレーションにより性能評価する。分散RAID4型の試作機では、所期の（サーバー数－1）に比例した性能向上を実現していること、及び1台のサーバーを突然故障させても、クライアント上で映像の乱れなく運転続行することを確認する。分散RAID5型についてはシミュレーションを実施し、所期の性能、即ち正常運転時には分散RAID4型よりも高性能であること、但しサーバーが1台故障した縮退運転時には分散RAID4型と同様の性能であることを確認する。次に分散RAID5型においてディスク故障の場合は、サーバー全体を切り離さず、ディスクだけを切り離す、ディスク規模の縮対機能を提案する。前からの単純な拡張でこれを実現した場合の問題点を指摘し、その解決策を示す。

第5章で、本研究を総括する。

目　　次

第１章　序論

１．１　研究の背景および目的

1935年アタナソフ教授によるABCマシンが、1946年にペンシルバニア大でENIACコンピュータが稼動して以来、50年間に渡るコンピュータの歴史の中で、その高速性の追求は常に繰り返し行われてきた。その１手段である並列処理技術についても研究の歴史は古く、1972年のILLIAC Ⅳが稼動して以来、1975年のC.mmp等種々様々な並列計算機が提案・試作されてきた。商用計算機では1970年代初めに大型汎用計算機IBM370シリーズにアタッチドプロセッサという形で２つのCPU（中央処理装置）のものが発表されて以来、4～8CPU構成のMP（Multi Prosessor）がメインフレームの最上位機として常にラインアップされてきた。この時代最先端の並列化技術はメインフレームが担ってきたが、今日これらはUNIXサーバーやさらに安価なパソコンサーバーに主役が受け継がれている。UNIXサーバーでは1996年に64ノード構成のExemplar Xクラス等の商用機が出ているが、パソコンサーバーはやや遅れており、1996年に初の8CPU構成のFT8000が発表された。これとは別に超並列マシンの分野では、コネクションマシンやプロセッサを8192個搭載したnCUBE2S等が商用化されている。

一方コンピュータ全般の技術進歩は、特にこの10年マイクロプロセッサを中心に急速な進展を遂げている。18ヶ月でトランジスタ数が2倍になるというMooreの法則にほぼ沿ったLSIチップの集積度向上がその原動力となっている。これによりスーパースカラやパイプラインのデカップル化等、高度なアーキテクチャの実装が可能となり、又マシンサイクルの高速化ともあいまって、マイクロプロセッサの性能は年率40%という驚くべき勢いで向上している。一方記憶系については大容量化が急速に進んでいる。特に磁気ディス

クは同様に 18 ヶ月で 2 倍の速度で容量が増大している。但しデータアクセス速度については、磁気ディスクがこの 20 年で Seek 速度が 3 倍、データ転送レートが 5 倍、DRAM は速度が年率 7%上昇といったように、進歩が遅い。この点がアーキテクチャ的工夫の材料を与えてくれる。

コンピュータ技術は上記のように激しく日進月歩の進歩を続けている一方、その骨格部分に目を向けると、その多くはほとんど変わっていないことに気づく。1946 年にフォンノイマンによって提唱されたプログラム内蔵方式は、そのまま現在に引き継がれている。命令セットについてもその基本構造が変わっていないのは周知の事実である。例えば最近話題の Pentium-Pro マイクロプロセッサに追加された MMX（Multi-Media Extension）命令群なども、その基本概念は一昔前の科学技術用ベクトル計算を高速化させる IAP（内臓アレイプロセッサ）用の付加命令セットなどと変わらない。

ハードウェアの基本構造についても同様で、CPU、主メモリ、磁気ディスクという基本構造はずっと変わっていないことに気づく。半導体メモリの集積度向上により、磁気ディスクの代替になり得るという期待を抱かせたことも何度かあったが実現せず、又フラッシュメモリや光ディスクにしても磁気ディスクの代替になるまでには到らず、それとは別の持ち場を得て普及している現状である。これは並列処理技術から見れば、扱う基本コンポーネントに変化がなく、研究の連続性が保たれてきた。

さて以上のような状況の下、コンピュータ性能は日進月歩で向上しているが、応用プログラム機能も同様に向上して要求 CPU 性能が上がり、その追いかけっことなっている。一方これまで非現実的と思われていた、膨大な処理時間を要するような応用についても現実的な視野に入ってきており、このためにマルチプロセッサによる並列処理を用いた更なる高速化の方法が追求されている。

さて、現段階で普及している並列処理技術は MIMD（Multiple Instruction stream Multiple Data flow）によるタスクないしスレッドを単位とした並列化である。即ち、1 つのジョブは複数のタスクに細分化され、複数のジョブが同時に並行して実行されるが、ある時刻で見た時には複数のタスクが同時に並行して実行されていることになる。各タスクは複数あるプロセッサ（CPU）のいずれかに割振られて実行される。複数ののプロセッサ間で主メモリとディスクは共有されており、1 コピーの OS の下に動作する。このため、

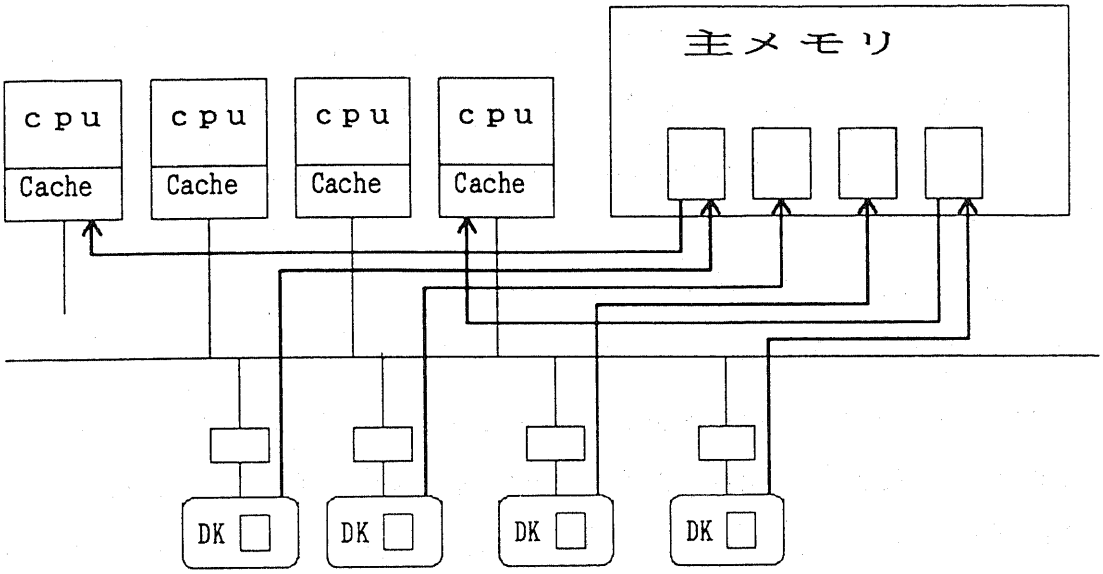


図 1.1　全共有方式マルチプロセッサ

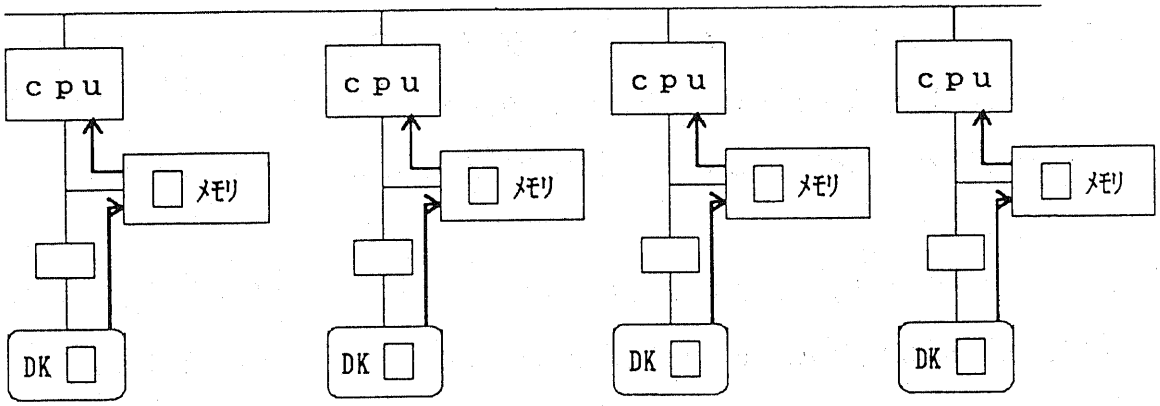


図 1.2　無共有方式マルチプロセッサ

あらゆるタスクはいずれのプロセッサ上で実行されることも可能である。このように複数のプロセッサがそれぞれ別々のタスクを同時並行実行して並列処理が実現される。この方式をタスク多重型の並列処理と呼ぶ。このためには必然的に、主メモリとディスクをプロセッサ間で共有する、いわゆる全共有方式（Shared Everything Architecture）のマルチプロセッサ構成が用いられる。この方式は、汎用機やUNIX機では1個のプロセッサの中身が実は4個のCPUから構成されているといったことは、高級機では今や当たり前という程度にまで普及してきている。

上記のタスク多重型の並列処理は、言うなれば比較的小さな多数の仕事を、1個づつ個々の人にバラまいて、並行実行させ高速化を図るものである。1つの仕事は1人の人が行い、これが終わると又別の仕事を行うといった形で進行する。これに対し1つの大きな仕事を多数の人が分担協力して実行する形の並列処理をタスク分割型の並列処理と呼ぶ。この方式については普及が遅れており、データベース等の一部の応用で実現されているに過ぎず、多くの研究課題が残されている。このようなタスク分割型並列処理は汎用化することは難しく、応用毎に特化した研究が行われている。本研究はこのタスク分割型並列処理を扱う。

タスク多重型の並列処理では1コピーのOS上で動くことから、必然的に全共有方式のマルチプロセッサが使われた。即ち主メモリとディスクが主体であり、その上に存在するOSから生成されたタスクはいずれのプロセッサが実行してもよい。この方式は、主メモリ競合のため、せいぜい8～16CPUまでが限度と言われている。図1.1は全共有方式のマルチプロセッサを示したものである。図に示されているように、この方式では複数のCPUが同一の主メモリにアクセスするため、CPU間のメモリ競合が起き、又複数のディスクも主メモリとの間で大量のデータ転送を行うためディスク間でもメモリ競合が起きる。このため並列度が増すと主メモリとそのバス上の競合が非常に大きくなり損失が増大するという問題がある。

これに対し図1.2は、プロセッサ間で主メモリとディスクを共有しない無共有方式（Shared Nothing Architecture）［Baru-95］のマルチプロセッサを示したものである。この方式では、それぞれのプロセッサやディスクはそれぞれのローカルメモリにアクセスするため、プロセッサ間ないしディスク間のメモリ競合が発生しないという大きな利点がある。このため原理的には並列度を際限なく高めることが出来る。そのかわり各プロセッ

サはそれぞれのローカルディスクにしかアクセス出来ないという大きな制約がある。このため基本的に、前もってデータを各プロセッサのローカルディスクに分割して入れておき、各プロセッサはそれぞれのローカルディスクに入っているデータについての処理を担当するという形をとる。このようにデータをいかに分散配置するかという問題は、一般に応用毎にそれぞれの戦略が存在するため汎用化は難しい。

このように無共有型マルチプロセッサの方が性能的にもスケーラビリティの点でも優れているが、応用毎にその並列化戦略が必要であるという制約がある。一方タスク分割型の並列処理を行うためには、タスク多重型並列処理と異なり全共有型マルチプロセッサである必然性はない。又タスク分割型並列処理の“応用毎に並列化の戦略が必要”という点は無共有型マルチプロセッサの制約条件と共通している。以上からタスク分割型並列処理には無共有型マルチプロセッサが適している。このため並列データベースマシンのようなタスク分割型並列処理を扱うものにおいては、ほとんどのケースにおいて無共有型マルチプロセッサ構成が用いられている［Teradata-83］［DeWitt-86］。

全共有型マルチプロセッサを用いたタスク多重型の並列処理は、ビジネス分野を中心に広く普及してきた。日々に生ずる多数の細かなトランザクションを、あたかも多数の窓口で処理するようにマルチプロセッサで処理される。一方これまで非現実的と思われていた、膨大な処理時間を要するような応用を対象とした並列処理はタスク分割型のものが多い。そしてそれは無共有型マルチプロセッサ上で行われることが多い。本研究ではこのタイプの並列処理を追求する。2 章ではデータベース応用を目的とした無共有型マルチプロセッサによるタスク分割型並列処理について、3 章と 4 章ではビデオサーバー応用を目的とした無共有型マルチプロセッサによるタスク分割型並列処理について論ずる。

グロシュの法則とは“コンピュータは価格が 2 倍になると性能は 4 倍になる”というものであるが、現代はこの逆、即ち“コンピュータは価格が 4 倍になっても性能は 2 倍程度”が真であると言われている［高橋-85］。筆者も、いろいろなアーキテクチャ的工夫をしても（原価向上）、払った努力程にはなかなか性能が向上しないことを何度か経験している。即ち逆グロシュの法則は、性能向上がいかに難しいかという事実を示している。そうであるならば安いコンピュータを多数使って高速化出来ないかという発想が湧き、これが本研究の一つの動機となっている。但しこれも容易なことではなく、n 台のコンピュータを使

えばｎ倍の性能が出るというような理想値は非常に難しい［高橋-85］。複数のプロセッサによるメモリ競合、通信、同期のオーバーヘッドにより、ある線を超えれば、プロセッサ数を増やしても最早それ以上性能が向上しない場合が出てくる。本研究では、このスケーラビリティの優れたマルチプロセッサシステムの方式を追求する。

さて本研究ではもう一つのサブテーマを設定している。以下にこれについて説明する。例えばリアルタイム OS を使ったシステムを開発する時、最近ではその開発チーム自身ではドライバーを作らなくなったと言われる。中核に近いドライバー開発は専門業者にまかせ、自身はもっぱら応用寄りの所を開発するそうである。以前はドライバー開発は、開発チーム内でも精鋭が開発担当した部分である。

さらには大手の OS ベンダーにおいてすら、その OS の基幹部分は外部にシフトする方向にあり、自身は応用（アプリケーション）プログラム開発に注力する、というような話も聞く。今やドル箱は応用ソフトに移ってきているからだという。

これらは技術の空洞化という視点からは必ずしも良いことではないが、とにかく潮流である。一昔前であれば OS とかドライバーといった中核部分は花形であったが、今ではブラックボックス化しており、ある意味で地位低下を来している。

コンピュータメーカが proprietary なハードウェアと OS で商売していた時代は去り、ハードウェアは Pentium を中心とした業界標準の LSI 部品で構成され、OS は Windows 系、または UNIX 系に統一されてきている。逆に言えば現在はそれらの市販品をうまく組み合わせて付加価値ある製品を提供する時代となった。

さて話を戻して“ある応用に対して高性能なコンピュータシステムを作りたい”という場合にも同じことが言える。ハードウェアやソフトウェアを 1 から作る時代は遠い昔のものとして過ぎ去り、今や出来る限り市販のハードウェアや OS や応用ソフトを利用して、これらを上手く組み合わせて作ることが重要となる。

このような手法を 1 つの技術としてとらえ、これを本研究のサブテーマとして考えることとしたい。

１．２　従来の研究の概要

１．２．１　データベースマシンの従来の研究

1970 年に E.F.Codd によりリレーショナルデータベースモデル［Codd-70］が提唱された後、1974 年頃“データベースマシン”という概念が登場してきた［植村-80］。それ以来今日に至るまで、種々様々なデータベースマシンが提案され、試作され、商用化されてきている。この一因としてはリレーショナルデータベース（RDB）処理は、その非定型処理で CPU 負荷が重く処理時間がかかるため、それを何とか高速化出来ないかという点、又 RDB の構造上ハードウェアフィルターとか並列処理といった、いろいろなの工夫が出来そうに見える点が挙げられる。又現在では世界標準となっている RDB 操作用の SQL 言語は、非手続き型言語でモジュール性に富み、少ない情報量で 1 まとまりの複雑なトランザクション処理を効率よく伝えるため、データベースマシンとのインタフェースとして優れている点も見逃せない。

データベースマシンの定義は“データベースシステム専用のアーキテクチャを有する計算機構”［植村-80］である。このためデータベースマシンというと、RDB 処理のための専用のハードウェアを種々備えているのが普通であり、時代を溯るほどこの傾向が強い。初期には“オンザフライ処理”と呼ぶ、ディスクから読み出したデータをその場で処理する方式が提唱された。トロント大学の RAP［Ozkarahan-77］が代表例であり、論理ヘッド（logic-per-track）と呼ばれ、ディスクヘッドの直後にフィルター機構他の演算機構を備える方式である。これに近いもので“知能ディスク”と呼ばれる CAFS［Babb-79］等がある。これは複数のディスクの直後にフィルタ機構やハッシング機構を備え、その場で処理することを特徴とする。RDB ではインデックス生成とか射影の処理等で大規模なソートが必要となるが、これを高速化するための各種ハードウェアソーターが提案、試作、商用化されている［Kakuta-85］［井上-90］。又、ジョイン演算を高速化するためにハードウェアのハッシング機構を設けるものとして“ハッシ化ビットアレー”［Babb-79］や、“Grace ハッシュ”［Kitsuregawa-83］等がある。又主記憶とディスクの中間に磁気バブルや CCD によるステージングバッファを設け、RDB 処理を主にこの上で行うようにして高速化を図る方式も EDC［植村-77］や DIRECT［DeWitt-79］等いろいろと提案、試作されている。

マルチプロセッサによる並列処理で高速化を図るものとしては、複数のプロセッサと複数の CCD ユニットと複数の大記憶をクロススイッチによって適宜接続する方式の DIRECT［DeWitt-79］や、数個～千個のプロセッサをＹネットと呼ばれる機構で結び、Ｙネット中をデータが通過する中で、ソートとかフィルタリングが行われるようにしたテラデータ DBC/1012［Teradata-83］や、ミニコン（VAX11/750）を 20 台 80Mbps のトークンリングバスに接続した試作機 GAMMA［DeWitt-86］他、多くの方式が提案されている。

以上のようにデータベースマシンと言えば RDB 処理専用のハードウェア機構を備えているのが普通であった。これに対し、ハードウェアは通常の計算機と変らず、単にソフトウェアを RDB 専用に作って高速化を図ったデータベースマシンとして、商用化して成功を収めたブリトンリーIDM500［IDM-85］がある。この意味では上記の GAMMA も、リングバスにミニコンを多数つないだだけという意味で、同じ分類に含めることが出来る。本論文で述べるデータベースマシン HDM（High-speed Database Machine）も同じくこの分類に含められる。即ち複数の、それぞれディスクを持つプロセッサを高速内部バスで接続したものであり、上記のような RDB 特有のハードウェア機構は全く備えていない。この意味で GAMMA に近いと言えるが、プロセッサ間を高速内部バスで接続する点、およびプロセッサ間通信に共有メモリを使う点等に構成上の相違が存在する。

さて無共有型マルチプロセッサによるデータベースマシンについて見た場合、データの分割・配置法が問題となる。［DeWitt-86］には次の 3 つの分類が示されている。

①ラウンドロビン

②ハッシング

③領域分割

③は例えば、10 代の人のレコードはプロセッサ 0 のディスクに、20 代の人のレコードはプロセッサ 1 のディスクに、・・というように、レコード中のキー値によって各プロセッサにデータを分配する。この方式は、例えば 20 代の人のレコードが特に多いとプロセッサ 1 のデータが多くなる偏りが起きるという、均等分配が難しい欠点がある。又例えば 10 代の人達を調べる場合、プロセッサ 0 だけが働くことになり、並列性が活かせないという問題がある。②はキー値をハッシングしてどのプロセッサに割り振るかを決めるもので、キー値が近いものをまとめる能力に欠ける。例えば“中村”と“中野”はハッシングすると全

く無関係な場所にマッピングされるため、“中～さん”を探したいというあいまい検索には適さない。 GAMMA や DBC/1012 を始めとして現代の多くの商用並列機は②ないし③を採用している。これに対し本論文で述べる HDM は①を採用している。この方式は均等分配性に優れるが、レコードは全くバラバラに配置されるという欠点がある。これに対して HDM では、クラスターインデックスによりこれらをまとめる新手法を適用した。これにより、キー値が近いデータをまとめて記憶し、かつ複数のプロセッサに均等に分散できるという願ってもない効果を得ることが出来た。この方式は 1991 年の米国等、3 ヶ国で特許成立しており、今後広く有償開放していく所存である。

１．２．２　ＶＯＤ技術に関する従来の研究

近年の CPU 能力の向上、メモリの大容量化、通信の発達、そして長年の研究の成果による画像（静止画・動画）圧縮技術の発達により、VOD（Video On Demand）技術の実現性が見えてきて、ここ数年急速な技術展開が進んでいる。とりわけ 1990 年に米国 GI(General Instruments) 社により 15Mbps のデジサイファー（Digicipher）方式が実演されて以来、映像のディジタル化が進んだことが大きなきっかけと言われている。

VOD システムは、multimedia storage server（メディアサーバー、本論文では直裁的にビデオサーバーと呼ぶ）に高速ネットワーク経由で多数のクライアントにつながり、ビデオサーバーから映像データを供給し、クライアントにて現状のビデオ再生装置と同様なビデオ再生が行えるものである。1 つ 1 つの映像データの流れはビデオストリームと呼ばれ、次のような特徴がある。

①実映像を乱れなく、即ちデータ供給の枯渇（starvation）を起こさず継続的に供給

②高速なデータ転送レートと大容量記憶が必要

このことから研究の対象は大きく、

①多くのクライアントに複数のビデオストリームを同時供給するビデオサーバー

②ビデオストリームを実時間映像として確実に伝送する通信路

の 2 つに分かれるが本研究は①に属する。

ビデオサーバーの考え方としては大きく以下の 2 つに分類される。

①データ供給の枯渇が起きないことを 100%保証するタイプ

②統計的に一定確率の下、データ枯渇が起きないことを保証するタイプ

①の例は、各ビデオストリーム毎にタイムスロットをきちっとスケジューリングして、その単位で確実にディスクからデータを読み出すようにした［Tobagi-93］がある。②の例では、通常のクラアントサーバー環境において、サーバーに少し手を加えただけでビデオサーバーにした［堂本-95］や［岡-95］などがある。本研究もこの②の分類に属するが、後述のようにこれで十分良質な映像が供給出来ることが実証される。

ビデオサーバーの中ではディスクアクセスがきわめて頻繁に行われるため、その効率化を目指した研究が多数発表されている。例えば個々のディスクの Seek 距離が最小になるような Scan と呼ばれるスケジューリング法［Reddy-93］が提案されている。又、ビデオストリーム群をグループ化し、グループ毎に順番に処理していくことにより、周期毎のビデオ供給のばらつきを減らすようにした GSS（grouped sweeping scheme）［Chen-94］や、同時供給中のビデオストリームを、ディスク内の物理的近さによってグループ化し、グループ内 Seek 時間を短くすることにより高速化を図る Region-based 法［Oyang-94］なども提案されている。又ディスクの台数を増やして、これに比例したディスク読み出しのバンド幅の実現を図る方法も種々提案されている［阪本-95］［内川-94］。

ディスク以外の所では、LAN の負荷変動に合わせて、符号化速度を変更する、即ち通信路が混んでいる時はビットレートの低い低品質映像を送るようにして、映像の途切れ（Starvation）を無くす方法が提案されている［岩見-93］。又音声の途切れを無くす目的で、回線の状態に応じて、音声伝送に適度な遅延を与える方法［田中-93］なども研究されている。これと関連して、ビデオサーバーではバッファ管理が重要な問題となる。前もってディスクからバッファーに多くのデータを読み出しておけば、一時的なディスク読み出しの遅延にも耐えられる訳であるが、これをまともにやると大量のバッファが必要となる。このためバッファリング手法に種々の工夫を加える研究も行われている［Tobagi-93］［Gemmell-94］。

以上の諸研究は部分的なところを対象としているが、本研究ではビデオサーバー自体の基本能力を高めることを目標とする。このために無共有型マルチプロセッサ構成を用いた分散 RAID 方式と呼ぶ手法を提案する。この方式は筆者等が独自に考案したものであるが、同じ考え方のものが分散ファイルシステムの分野で既に試作評価されている［Cabrera-

91] [Long-94]。しかしながら、この分散 RAID 方式をビデオサーバーに適用して試作評価したのは筆者等が最初であり、この点で多くの新しい成果を得ることが出来た。

１．３　本研究の特徴

データベース応用において、定型的トランザクションを時間当たり多数処理すればよいのであれば、現状行われているような、タスク毎に個々のプロセッサに割り当てて実行するタスク多重型の並列処理方式で十分である。ところが、非定型なトランザクション、例えば 100 万件規模のデータベースで、インデックスが利用出来ないジョイン処理を行う等の、1 時間もかかるような大きなトランザクションを実行する場合、この方式ではそのタスクはどれか 1 つのプロセッサに割当てられて実行されるため、並列処理による高速化が図られないという問題がある。この場合には 1 つの大きなタスクを複数に分割して複数のプロセッサに割当て分担して実行させ、高速化を図る必要がある。即ちタスク分割型並列処理が必要となる。

一方ビデオサーバー応用についても同様のことが言える。1 台のビデオサーバーにつながるユーザ数が増えてきて、これから自然に同時配信するビデオストリーム数が増えて、サーバー能力の限界に達した時、ビデオサーバーをもう 1 台増設してビデオプログラムの半分をこちらに移すとする。サーバー1 台の同時ビデオ配信数を v とすれば、2 台に増えれば仕様上は確かに 2v に増強される。但しこれは 2 つのサーバーが保持するビデオ群に丁度要求が均等に来た場合の話である。2000 本のビデオタイトルがある時、その上位 10 本に 80%の要求が集中するそうである。サーバー数が n 台と増えればこのような要求の均等分散は一層難しくなるのは明白である。これが均等化されないと、あるサーバーにだけ負荷が集中して他のサーバーは遊んでいる状態になり、システム全体のスループットは低下する。これを解決するには、1 本 1 本のビデオプログラムの供給を複数のサーバーで均等に分担して行えればよい。

以上のような問題を解決するために、本研究では無共有型のマルチプロセッサ構成によるタスク分割型並列処理を研究する。これの場合応用に特化した並列化戦略が必要となる

が、一つはデータベース応用を対象とし、他の一つはビデオサーバー応用を対象とする。それぞれの応用について、最適な並列処理システムを構築することを目指すが、これらに共通する特徴は以下の通りである。

［特徴］

・無共有型マルチプロセッサ構成を用いる

・高性能化を追求する

・価格性能比の向上を追求する（安価なプロセッサを多数使用－逆グロシュの法則）

・スケーラブルな性能向上を目指す

・応用に専用化することで高速化を図る（ex. RDB 処理向けディスクアクセス方式）

・試作を行いエンドユーザが評価できるレベルでの検証を行う

・市販のハード／ソフトを極限利用することにより、開発費の低減を図る

１．４　論文の構成

本論文は全５章からなる。

第１章は序論であり、研究の背景と目的、これまでに行われた他の研究の概要を述べ、本研究との関連を示した。又、本研究の特徴について述べた。

第２章では、リレーショナルデータベース（ＲＤＢ）処理の高速化を目的とした、無共有型マルチプロセッサによるデータベースマシンを提案し、試作結果について述べる。ハードウェア構成、ソフトウェア構成上の特徴、その上での並列化アルゴリズム、ＲＤＢに特化したディスクアクセス手法、ディスクキャッシュ構成法、そして障害回復法の特徴について記述し、最後に拡張ウィスコンシンベンチマーク等による性能評価結果と、このマシンの特徴を生かした適用事例について述べる。

第３章ではビデオサーバーの高性能化（ビデオ配信数増大）を目的とした、無共有型マルチプロセッサシステムの提案と試作を行う。これに先立ちＬＡＮ上でビデオデータ転送の評価実験を行う。この結果を反映して、通常のＲＡＩＤディスクアレイの、ディスクをサーバーに置き換えた形でマルチプロセッサ構成とする、分散ＲＡＩＤ０型ビデオサーバーと呼ぶ方式を提案する。この試作を行い、特に所期の目標であるサーバー台数に比例し

たスケーラブルな性能向上が実現出来ているか否か、映像品質が良好であるか否か等を重点に評価を行う。

第４章では性能に加えて高信頼性も追求した分散ＲＡＩＤ４及び分散ＲＡＩＤ５型ビデオサーバーと呼ぶ方式を提案し、それぞれ試作ないしシミュレーションにより性能評価する。分散ＲＡＩＤ４型の試作機では、所期の（サーバー数－１）に比例した性能向上を実現しているか否か、及び１台のサーバーを突然故障させても、クライアント上で映像の乱れなく運転続行出来るか否かを重点に評価する。分散ＲＡＩＤ５型についてはシミュレーションを実施し、所期の性能、即ち正常運転時には分散ＲＡＩＤ４型よりも高性能であるか否か、サーバーが１台故障した縮退運転時には分散ＲＡＩＤ４型と同様の性能であるか否かを評価する。次に分散ＲＡＩＤ５型においてディスク故障の場合は、ディスクだけを切り離す、ディスク規模の縮対機能を提案し考察を加える。

第５章では、本研究の成果についてまとめ、今後の課題について述べる。

第２章　並列データベースマシンＨＤＭの提案と開発

本研究のテーマはＣＰＵとディスクを複数用いて高速化を図るアーキテクチャの研究であるが、この裏に底流として、出来るだけ世の中に既に存在する市販の標準部品を利用してシステムの開発費の低減を図るというもう一つのサブテーマが存在する。本章では、CPUとディスクを複数用いて高速化を図ることを特徴とするデータベースマシン HDM (High-speed Database Machine) について述べる。従来提案された多くのデータベースマシンがそうであったようにデータベース処理の高速化のため種々の高速アルゴリズムを実現する専用ハードウェアを備えるということをせず、市販されている標準部品だけで構成して開発費の削減を図ったのが特徴である。

リレーショナルデータベース（RDB）は CPU 負荷が重いため実行速度が遅いと言われるが、この点を解決するために CPU のみならずディスク入出力の高速化も図るべく、ディスクも併せた並列処理を実現して、リレーショナルデータベースを高速に実行するデータベースマシンを開発した。このシステムは複数のソースから商用化され、現在多くのサイトで稼動している。今日 RDB はデータベースの主流となり多くのシステムが市販されているが、その動作形態はインデックス（索引）を利用した定型処理が中心である。（インデックスを利用した前方一致検索のように、一定あいまい検索（非定型検索）的なことも出来るため誤解しやすいが。）数 10 万件レベルのデータベースにインデックスの利用出来ない非定型検索を現状の市販システムに実行させると、10 分以上待っても答えが返ってこないのが普通であるが、HDM では 1 分程度で答えが返ってくる。このように HDM は種々のアーキテクチャ的工夫を加えることにより高速性を達成している。

２．１　緒言

企業活動の情報処理化が進み、各種のデータが大量に集積・保存されてきている。このデータを統計／分析などで活用し、事実に基づく判断を行いたいという要求は非常に高い。しかし、従来の技術では大量データを対象とした自由な検索などのデータ処理は、

①高価になりすぎる

②応答時間が何時間／何十時間と著しく悪い

等、現実的には実用的なシステムができず、データはデッドストックとなっていた。本研究はこれらの要求と現実のギャップを埋め、大量データ検索の実用的システムを提供し、あらゆる分野でのデータ活用を手軽に行えるようにすることを目的とする。

本研究で開発した技術の特徴は以下の通りである。

（１）高速検索マシンによる高速データ検索技術

①集積された大量データを自由に活用するためには、索引（インデックス）付き項目などに限定することなく条件を設定できる必要がある。このような非定型検索を実行するためにはすべてのデータをスキャンして目的のデータを探し出す必要がある。これを従来の速度の 10 倍を超えて高速に実現するために、ディスク入力を含む全ての処理を並列に処理する並列処理ハードウェア、及びそれを制御する並列処理リレーショナルデータベース管理ソフトウェア技術を開発した。

②４台の各 CPU の下でデータベース管理ソフトウェアは独立して動作し、受け取った検索要求を並列に処理する。

③各々の処理結果を連携して検索結果として応答する。各 CPU の処理時間が均一となるように、データは各 CPU のディスクに均等に格納する。

④ディスク入出力速度を最大限に引き出すため、入出力サイズを大きくとり、非同期で連続入力する。

（２）低価格化のための実用化技術

低価格化のために徹底した汎用部品の採用を行い、量産効果によるコストダウンを可能とする。

（３）標準化のための実用化技術

大量データの活用は各々の部門の専門家によって行われる。そのためその人達が使いこなせる容易性と、既存知識が生かせる標準性が必要である。このため標準のパソコン、Ethernet LAN、TCP/IP といった、各部門で容易に利用できるデファクトスタンダードな技術を組み合わせることで実現する。

この結果、従来技術と比較して以下のような優れたデータベース検索システムが開発出来た。即ち、このシステムでは、100 万件規模のデータの自由な項目（アトリビュート）に選択のための条件を設定し検索を行っても 1 分程度で応答し、従来の汎用機やオフコン、サーバー等でデータベース管理ソフトウェアを利用するのと比べ非常に高速である。又価格は数百万円～1 千万円程度と、同一規模のデータ処理を行う従来のハードウェア及びソフトウェアの価格と比較すると非常に低価格である。

２．２　ＨＤＭの設計思想

よく知られているように、リレーショナルデータベース（RDB）オペレーションは並列処理に向いている。それ故高性能な 32bit マイクロプロセッサを複数用いて並列に動作させることにより、高性能データベースマシンが出来るのではないかと考えた。誰がどんなプログラムを投入するかもわからないようなケースを対象とした、一般的な並列処理はなかなかむつかしい問題である。これに対し、データベースマシンの場合要求されるのは RDB オペレーションだけであるため、前もって RDB 処理をいくつかに分類しておき、それぞれのケースに対していかに処理を並列化させるかを決めておくことができる。

他方、現代の計算機におけるファイルのアクセスメソッドは、OS の主要部を占める程大きくなり高機能化しているため、ディスクはユーザーからははるかに遠い存在となっている。汎用的なアクセスメソッドを使わず、直接ディスクを制御したらデータベース処理に専用化した、かなり効率的なディスクアクセスが行えるはずだと考えた。

以上を考慮し、以下のような基本的設計思想の下に設計をおこなった。

（１）並列処理を適用した RDB 処理

（２）H/W と S/W を RDB 専用に作った

（３）マイクロプロセッサと DRAM をふんだんに使用

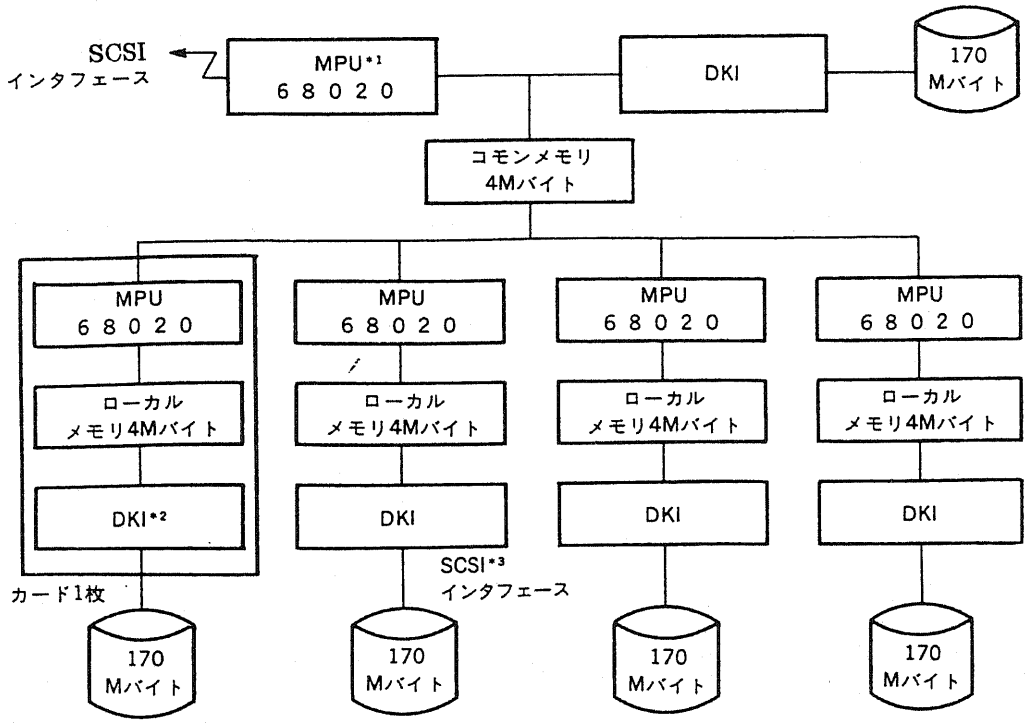
（４）ディスクアクセスメソッドを RDB に最適化

（５）バス方式を採用した中規模でコンパクトな性能価格比を重視したマシン

（６）全面的に標準部品を使用

（７）アクセス言語として SQL を採用

上記（５）は HDM にとって非常に特徴的な要素となっている。即ちバスで複数のマイクロプロセッサをつないでいるために、現実的なプロセッサの最大数は 17 個程度に押さえられる。（バスリピータを付ければこの数倍程度の個数までは可能であるが、バスネック等により性能向上率が非常に悪くなる。）それ故、プロセッサ数を 1024 個まで増やせるテラデータ DBC／1012 のように超大型のデータベースマシンにはなりえない。その代わりバス構造であるためハードウェアは非常にコンパクトにできる。又バス自体も非常にシンプルなものであるため、ハードウェア量が少ないと同時に、データ転送時の時間的ロスも少



注　＊1 MPU : Micro Processing Unit　＊2 DKI : Disk Interface Logic
　　＊3 SCSI : Small Computer Systems Interface

図 2.1　HDM 試作機のハードウェア構成図

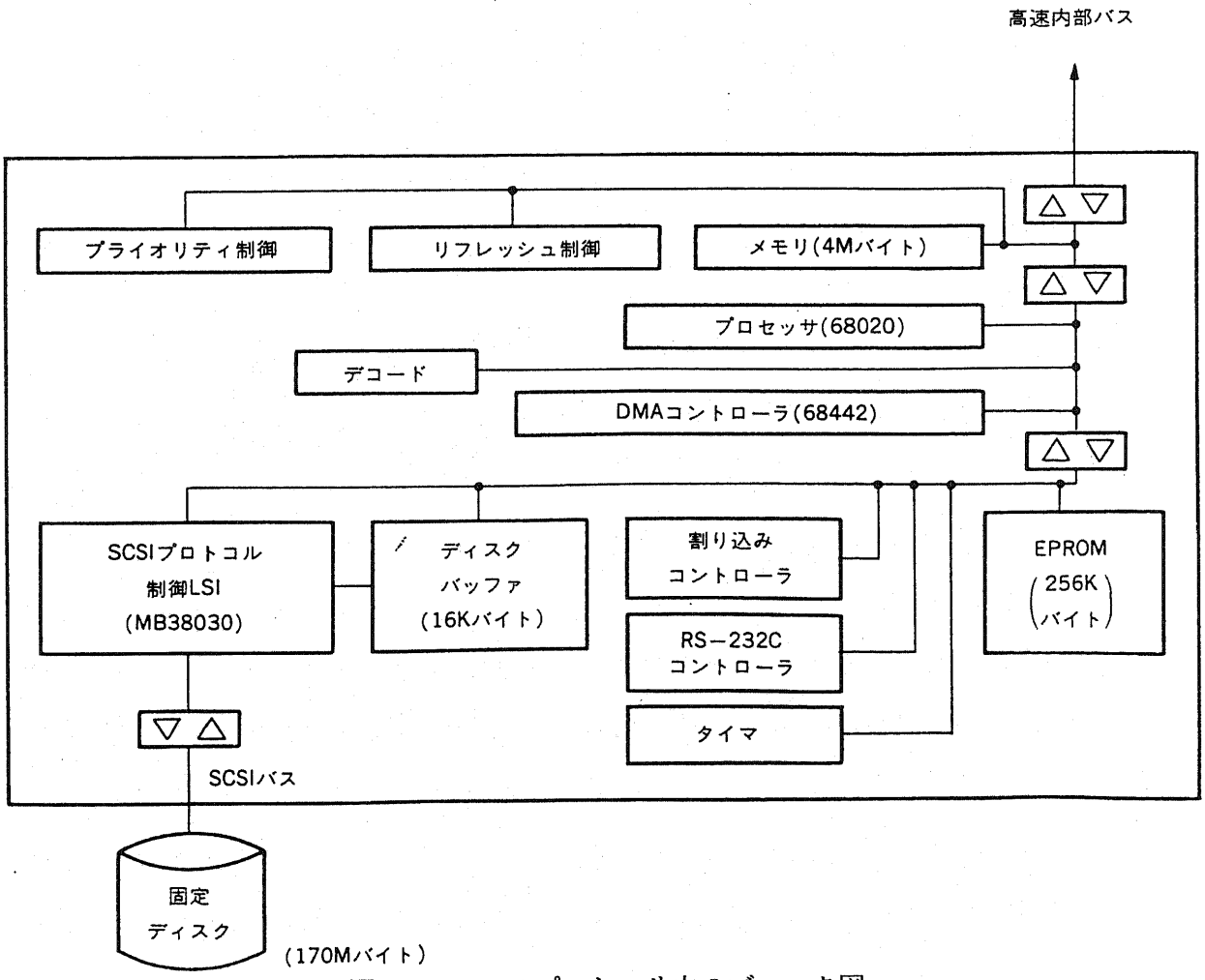


図 2.2　一つのプロセッサ内のブロック図

ない。

一般にバス方式は最も効率の良いハードウェア形態であるといえる。これは一つのデータ転送路を複数のユニットがシェアーして使うからであり、又バス競合による性能低下もトータルな処理時間から見れば大したことはないことが多い。

２．３　ハードウェア構成

図 2.1 に HDM 試作機のハードウェア構成を示す。図に示されるように HDM 試作機は１つのマスタプロセッサと４つのスレーブプロセッサで構成される。１つのプロセッサは１枚のカード（30cm 四方）から成り、５枚のカード（マスタ、スレーブ共通。ROM 内容だけが異なる。）が単純な高速バス（10Mbps）で接続されている。このほかにプロセッサボードに SCSI インタフェースで接続する複数のディスク装置と電源があるだけである。

HDM ではこのようにプロセッサ間をバスで接続したところに大きな特長があり、ハードウェア量の削減につながっている。図 2.2 に各プロセッサの中味のブロック図を示す。プロセッサ間の割り込み制御機能と高速バスを介しての共有メモリアクセス機能を除けば通常のマイクロプロセッサボードと同様の構成である。ただし、ディスクがデータを転送中でもＣＰＵがフル回転できることがこのマシンの命であるため、市販のＤＭＡの性能不備を補うべく、ディスクバッファロジックを追加してディスクのデータはバースト転送するようにした。

各プロセッサは 4M バイトのローカルメモリを持っており、マスタプロセッサのメモリは共有メモリとしても使われる。共有メモリとローカルメモリのメモリマップを図 2.3 に示す。図 2.4 と図 2.5 は共有メモリとローカルメモリのアクセス法を示す。図 2.4 は通常のメモリアクセスサイクルを示しており、マスタプロセッサは共有メモリ（コモンメモリ）に、各スレーブプロセッサはそれぞれのローカルメモリにアクセスする図である。これに対し図 2.5 はスレーブプロセッサの１つ（最下段のプロセッサ）が共有メモリにアクセスするメモリサイクルを示している。この場合には他のスレーブプロセッサはそれぞれのローカルメモリにアクセス可能であるが、マスタプロセッサは共有メモリにアクセス出来ない。このためスレーブプロセッサの共有メモリ使用要求が頻繁に生じた場合でも、マスタプロセッサが共有メモリを使用できるように、図 2.6 に示すような制御を行っている。即

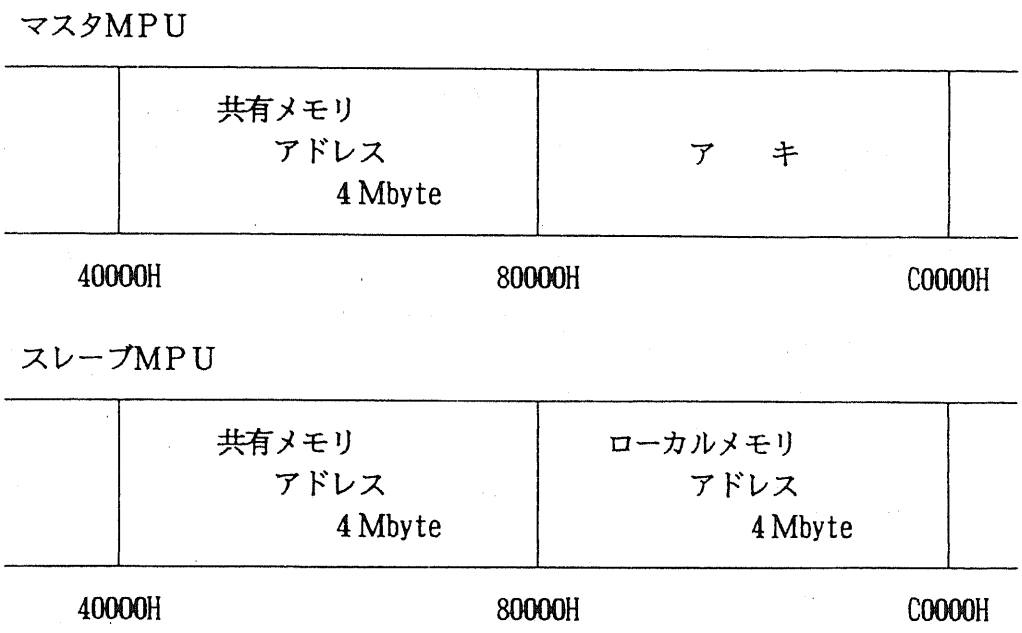


図 2.3　メモリマップ

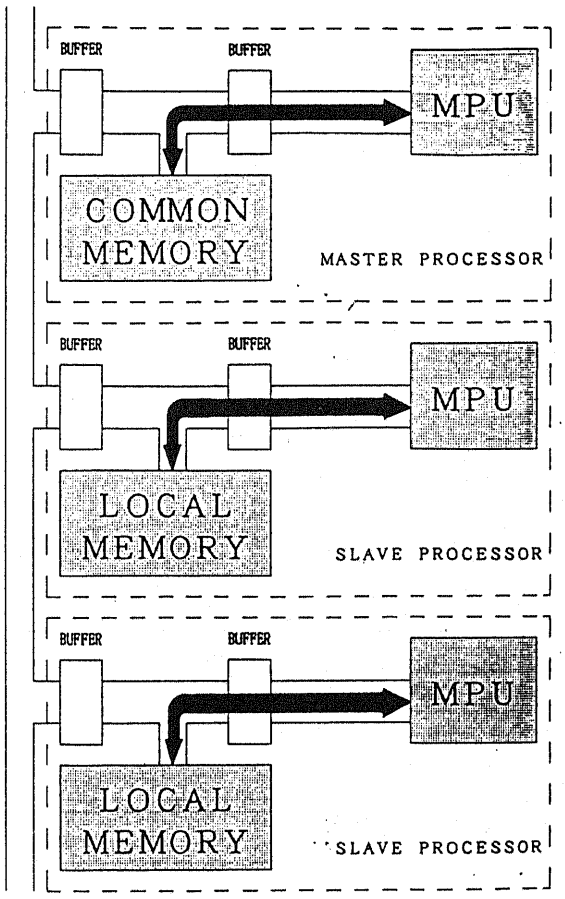


図 2.4　メモリのアクセス方法

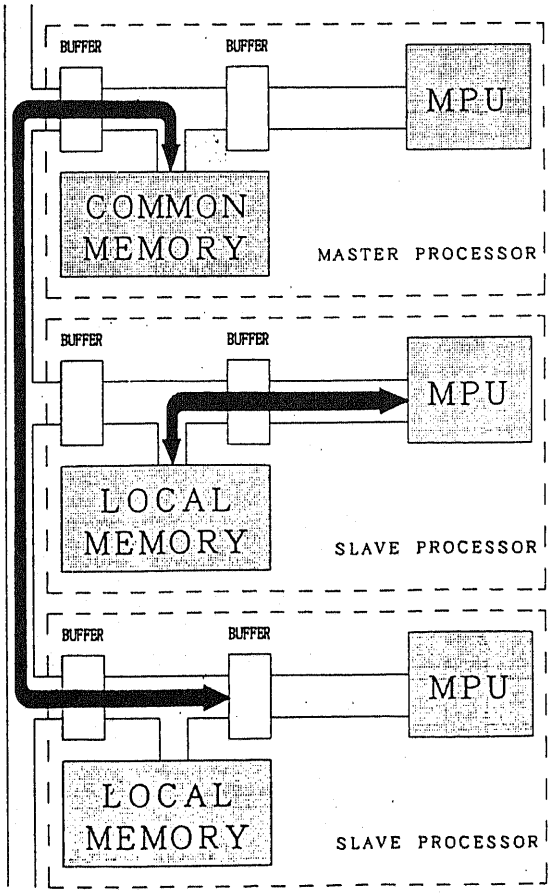


図 2.5　共有メモリのアクセス方法

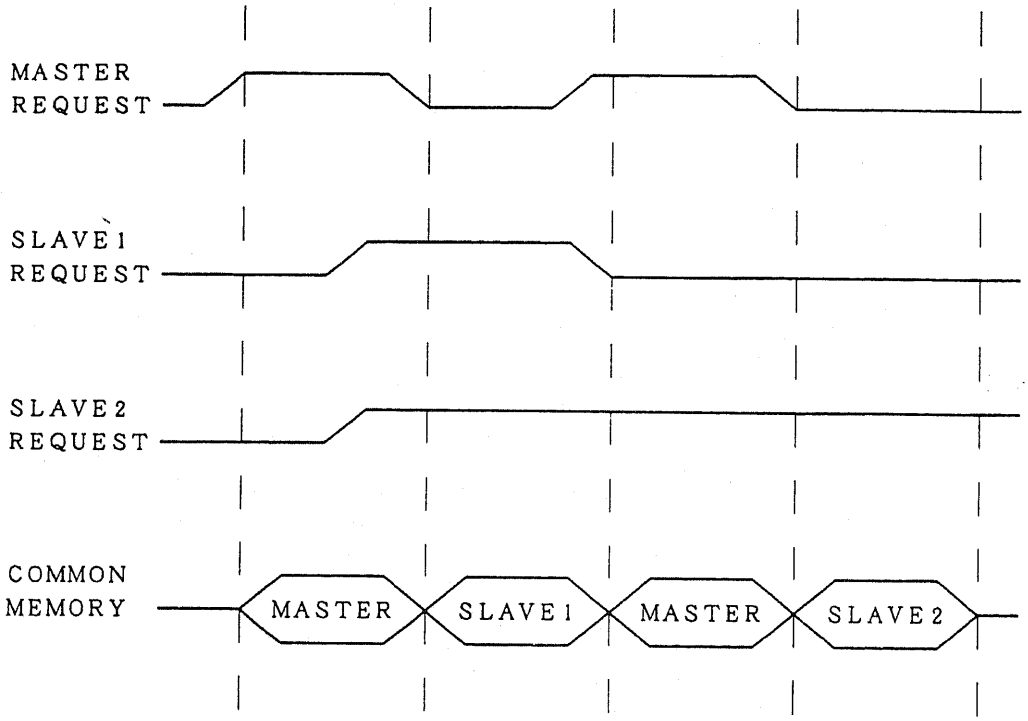


図 2.6　コモンメモリアクセス

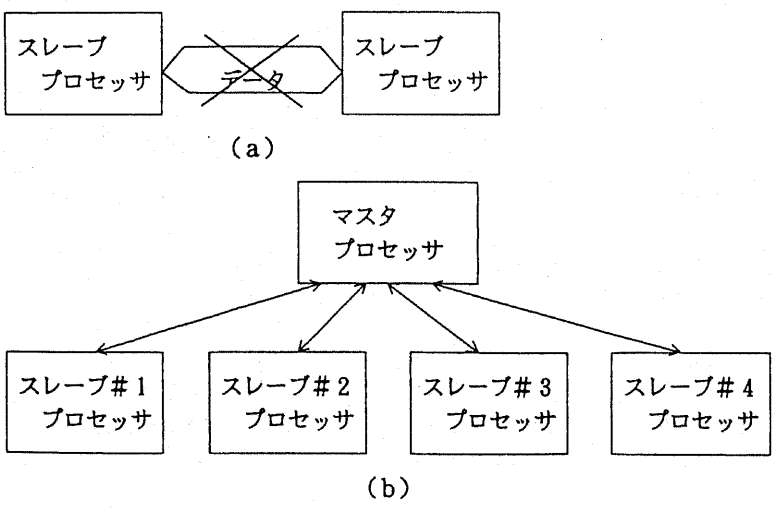
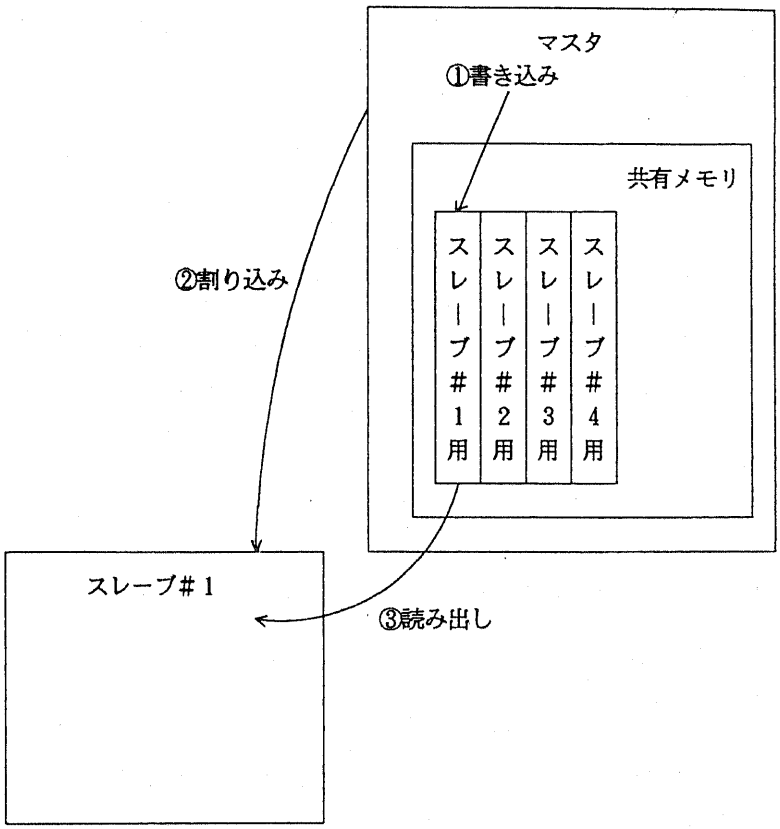


図 2.7　プロセッサ間の通信



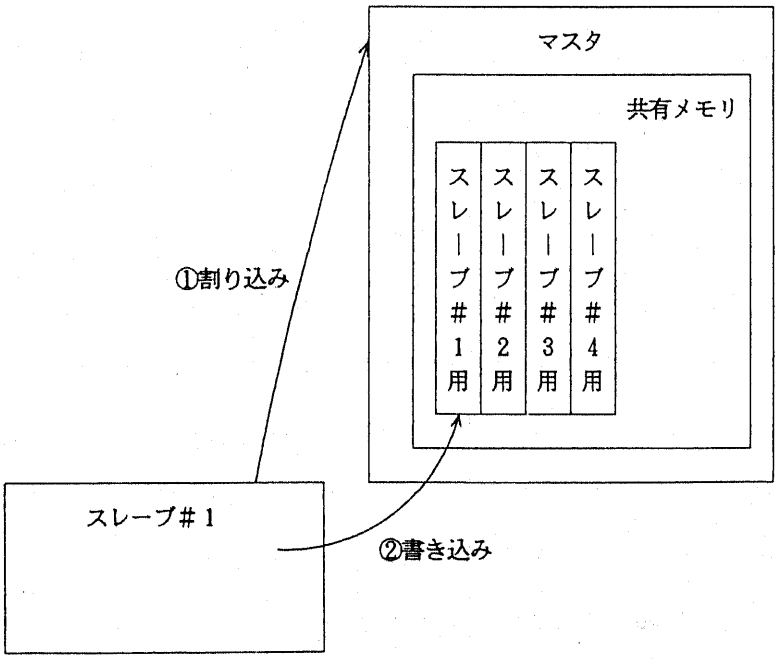


図 2.8　マスタとスレーブの通信

ちスレーブプロセッサとマスタープロセッサは必ず1回ずつ交代して共有メモリを使用するように制御している。

一方、マスタプロセッサと各スレーブプロセッサの間には双方向の割り込み機能があり、これとコモンメモリによってマスタスレーブ間通信が行われる。マスタープロセッサからスレーブプロセッサへの通信法を図2.8（上）に、スレーブプロセッサからマスタープロセッサへの通信法を図2.8（下）にに示す。このようにプロセッサ間の通信はマスタープロセッサとスレーブプロセッサの間だけで行われ、スレーブプロセッサの間では通信は行われない（図2.7）。

以上のようにHDMではデータベースマシン特有のハードウェアは持たず、32ビットマイクロプロセッサの高速処理性能に頼り切っており、それを単純な高速バスでつなぐというアーキテクチャを取っている。

以上がHDM本体ハードウェアであるが、この他にフロンとエンド機としてUNIXワークステーションが接続され、全体としてネットワーク上のデータベースサーバーが構成される。UNIXワークステーションはマスタープロセッサとSCSIインタフェースにより接続され、2.5.1で述べるようにSQLの中間言語が論理的インタフェースとなる。フロントエンドのUNIXワークステーションの最小限の機能は、LAN上でのTCP/IP通信を行い、各クライアントからのリクエストを受け付けるサーバー機能を実現することである。クライアントからのアクセス形態には、クライアント／サーバー型のものと、リモートログインによる集中型のものの2種類がある。前者では2.5.1で述べるSQLパーサーやSQIプログラムは各クライアント・ワークステーション上に搭載され、RPC（Remote Procedure Call）により要求が行われる。この場合にはフロントエンド機は上記の最小限の機能のみを実行する。一方後者の場合にはSQLパーサーやSQIプログラムはフロントエンドのUNIXワークステーション上に搭載され、フロントエンド機はSQL言語の解析やユーザーインタフェースも含めて実行する。この場合はクライアントは単にリモート端末として動作する。いずれの形態をとるかは、適用システムの事情に合わせて選択される。

２．４　ＨＤＭのアルゴリズム

２．４．１　データベースの水平分割

HDM ではデータベースに対する処理を並列化するために、１つのリレーションのタプル群をそれぞれのスレーブプロセッサに均等に配分して持っている。(図 2.9) この手法は水平分割方式と呼ばれている [DeWitt-86] 。このようにリレーションを均等に分割された状態を保ちながら、タプルを格納していくために次のような手法を用いている。

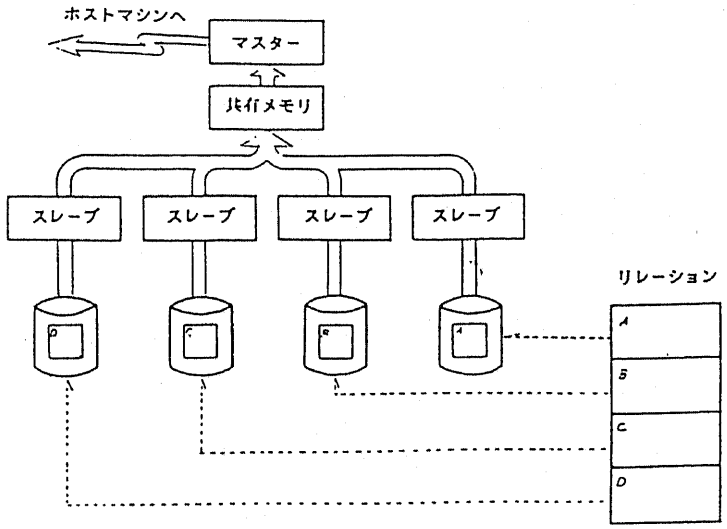


図 2.9　水平分割されたリレーションの全件サーチ

①もしリレーションがクラスタードインデックスを持っている場合には、タプルが格納される時に、それは決まった特定のスレーブプロセッサのディスクに格納される。即ち、挿入されるタプルは、そのクラスタインデックスのキー値に従って決定される所定のページに格納されるため、そのページを格納しているスレーブプロセッサに格納されることとなる。この時もしそのページが既に満杯であるならばページスプリット [Ullman] という処理が行われる。即ちそのページの内容は２つのページに分割され、そのうち一方のページは、その時点でそのリレーションのタプルの数が最も少ないスレーブプロセッサのディスク上に送られて格納される。

②そのリレーションがクラスタードインデックスを持っていない場合には、単にタプルをそのリレーションのタプルの数が最も少ないスレーブプロセッサのディスクに送って、そのディスク中に格納する。

注）この手法は筆者が発案したものであるが、日、米、独の３カ国で特許取得しており、商用システムにおいてリレーショナルデータベース処理の並列化が盛んに行われるようになってきた今日、きわめて有効な特許となってきている。このため今後有償開放等の有効利用を図る考えである。(米国特許 5058002 号(1991 年) PAGE SPLITTING METHOD AND APPARATUS FOR A DATABASE STORED IN A PLURALITY OF MEMORY STORAGE UNITS, S.NAKAMURA etal..独特許 3821551 号（1994 年）Daten verar beitungs vorrichtung, S.Nakamura etal..)

２．４．２　トランザクション処理の基本

トランザクションを処理するために必要なリレーションのスキーマ情報はマスタープロセッサとスレーブプロセッサが、それぞれ必要な分を分担して持っている。クラスタードインデックスおよび非クラスタードインデックス自体はマスタープロセッサが、リレーションのデータ自体は各スレーブプロセッサが持っている。マスタープロセッサは、トランザクションをスキーマによってチェックし、必要ならインデックスを調べた後、スレーブプロセッサに制御コマンドを送る。このとき、データの処理範囲をインデックスを用いることによって限定できるなら、そのインデックスが持つページ番号も渡される。インデックスが使えないときは、スレーブプロセッサは全件サーチを行う。

２．４．３　ジョインの処理方式

基本的には、HDM のジョイン処理ではインデックスが用いられる。（図 2.10）ジョインを行うアトリビュートがインデックスを持っていない場合にはジョイン処理を行う前に一時的にインデックスを生成する。以下にジョイン処理の概要を示す。

①マスタープロセッサは、各リレーションからジョインを行うアトリビュートのインデックスをディスクから取り出し、各要素を順に比較していくことによってインデックスのジョインを行う。

②マスタープロセッサはジョインされたインデックスからタプルの位置情報を取り出し、各スレーブプロセッサへタプルを取り出すためのコマンドを発行する。

③各スレーブプロセッサはディスクからタプルを取り出し、それにプロジェクション処理を施した後、それを共有メモリ上に作られたキャッシュファイル上に置く。

④マスタープロセッサはこのキャッシュファイル上にあるタプルをジョインする。

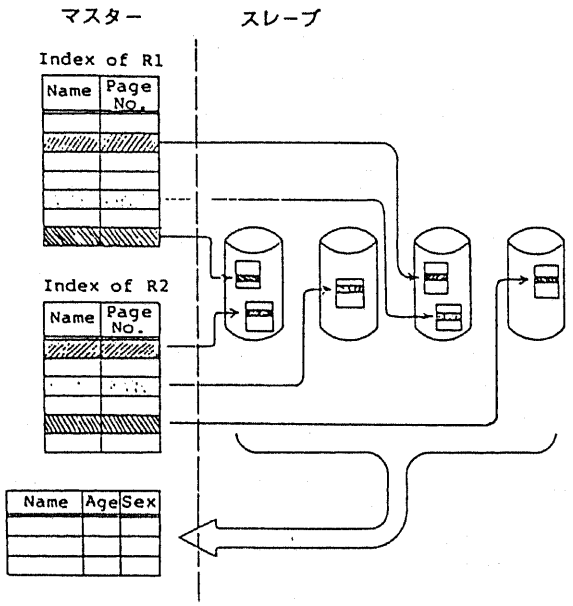


図 2.10　インデックスを用いたジョイン

２．４．４　処理の並列化

HDM で並列化によって高速化される処理の代表的なものを以下に示す。

①全件サーチの高速化

HDM ではリレーションを水平分割して格納しているので、インデックスが使えなくて、全てのタプルをサーチしなければならないというような全件サーチにおいて、各々のスレーブプロセッサが並行してサーチ処理を実行して処理が高速化される。（図 2.9）

②並列ソーティング

HDM では各スレーブプロセッサが並行してデータをそれぞれのディスクからローカルメモリに読み込むと同時に、並行してクイックソートを実行する。ソートされた結果は共

有メモリを介してマスタープロセッサに渡され、マスタープロセッサがこれらのデータをマージしていく。この様にスレーブプロセッサが並行してソート処理を実行するため、全体のソート処理時間が短縮される。

③多重トランザクション処理

データベースでは、定型業務として、通常１タプルに対する更新、追加、削除などの処理が頻繁に行われる。これらの処理はいずれか１つのスレーブプロセッサのみによって処理される。このような処理が同時に多数要求されたとき、それらは確率的に見て各々のスレーブプロセッサにばらつくと考えられる。そこで、複数のスレーブプロセッサの間でこれらの処理が並行して実行されて高速化が図られる。

④限定された範囲での全件サーチ

並列に処理するとは言っても、全件サーチ自体は非常に高負荷な処理である。しかしながらクラスタードインデックスによってサーチの範囲が限定されることが多いため、全件サーチはリレーションのある限られた範囲に対して行うだけでよいことが多い。この時、クラスタードインデックスにより、その範囲に含まれるタプルは物理的ないくつかのページにまとまって格納されている。そのようなページは、先に述べたリレーションの水平分割化格納方式を用いていれば全てのスレーブプロセッサ間に散らばっている。そのためこのような範囲が限定された全件サーチの場合でも、それぞれのスレーブプロセッサで並行して処理が実行される。（図 2.11）

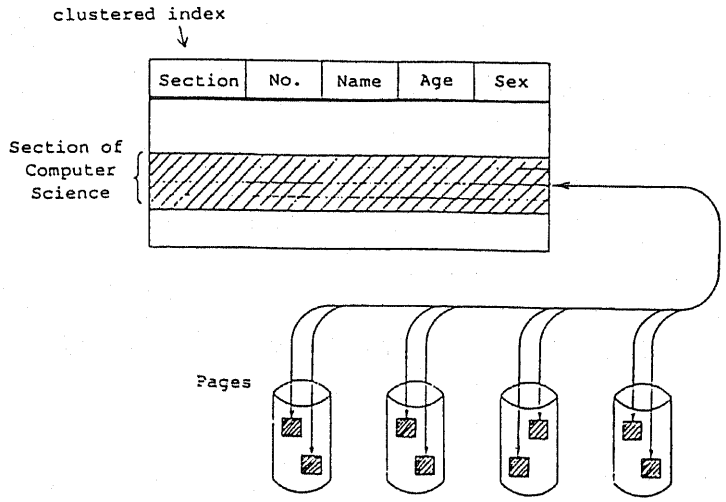


図 2.11　限定された範囲での全件サーチ

図 2.11 の例は大学の学生のデータベースの例である。各学生の情報がそれぞれ 1 タプル（1 レコード＝1 行）に記憶される。この例では Section（学科）のアトリビュート（項目＝列）がクラスタード・インデックスに指定されている。そのためこれらのタプルは Section のアトリビュートでソートされ、物理的にもこの順に並べられている。（あくまでページ内で。ページのレベルでは物理的にバラバラに配置されている。）ここで図 2.11 の例のように、“コンピュータサイエンス”の学科の学生という条件が与えられた場合、それらのタプルが存在するページは全体のページの中の部分集合に限定される。図 2.11 に図示された例によれば 8 ページに限定され、2 ページずつ各スレーブプロセッサのディスクに記憶される。各スレーブプロセッサは、これらの 2 ページに対して全件サーチを行えばよいため、全体に対して全件サーチを行う場合に比べて高速化される。

⑤インデックスを用いたジョイン

インデックスが利用できる場合のジョイン処理では、まずこのインデックスを用いてジョインされるタプル群を求める操作を行うが、この後これらのタプルを実際に読み出す時に、各々のスレーブプロセッサのディスクから並行して読み出すことが出来る。（図 2.10）

図 2.10 の例では左から 1 番目と 3 番目のスレーブプロセッサは 2 タプルを、左から 2 番目と 4 番目のスレーブプロセッサは 1 タプルを読み出せばよい。このようにタプルの読み出しが並行して行われるため高速化される。

以上のような並列処理のアルゴリズムを処理系に実装して、高速なリレーショナルデータベース処理を実現している。

２．５　ソフトウェア構成

２．５．１　概要

HDMのソフトウェアは、次の点に留意して設計されている。

①汎用的なユーザインタフェースを提供。

②ホスト上ではリレーション情報を持たないようにし、ホストの負荷を軽くすると共にポータビリティを増す。

③ハードウェアの特長（マルチプロセッサ、大容量メモリなど）を生かす。

④データベース処理の専用ソフトエアに徹する。

図2.12に示すとおりHDMソフトウェアはホストマシンとマスタプロセッサ及びスレーブプロセッサ上で動作するもので構成される。ホストマシン上には、SQL言語を中間言語に変換するSQLパーサーとユーザーにインタラクティブなインタフェースを与えるSQI（Simple Query Interface）がある。このようにデータベース処理言語として、SQLを採用することで、汎用的なユーザーインタフェースを提供した。

マスタプロセッサはホストマシンからSQLを中間言語に落とした形式で、処理要求をトランザクションとして受け取る。マスタプロセッサ上のプログラムは、スレーブプロセッサを並列に動作させて、1つのトランザクション内における並列処理とか、複数のトランザクション間の並列処理を実現している。

トランザクションを解析し、その処理を実行するのはトランザクション処理プログラム（TPP）である。TPPが動作するタスクはいくつか用意されていて、それらは並行に動作し、各々が別々のトランザクションを処理する。TPPはリレーションの実際の操作等をスレーブプロセッサ通信プログラム（SCP）経由で、プロセッサコマンド処理プログラム（PCP）に要求を出す。一つのTPPが同時に複数のスレーブプロセッサに要求を出すと、一つのトランザクションの処理が各スレーブプロセッサ上で並列に実行される。一方、TPPが各々別々のスレーブプロセッサに別の要求を出すと、各々のトランザクションの処理が各プロセッサ上で並列に実行される。このようにマルチプロセッサを有効に動作させる仕様となっている。

キャッシュマネージャー（CM）とキャッシュファイルマネージャ（CFM）は、大容量

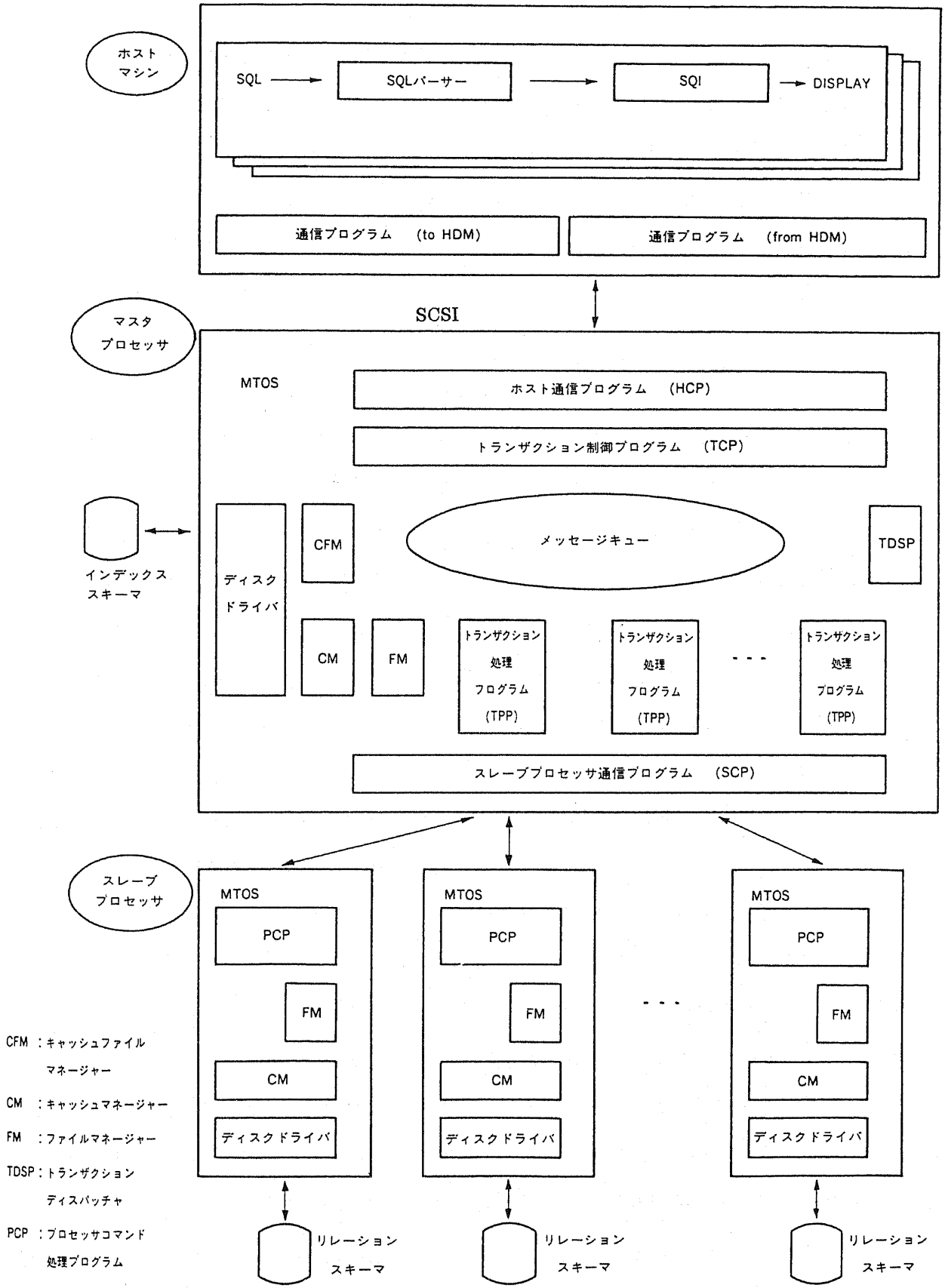


図 2.12 HDM ソフトウェアの構成

メモリをディスク入出力バッファとして使う高速処理機構を提供する。

スレーブプロセッサ上のプロセッサコマンド処理プログラム（PCP）はマスタプロセッサから要求を受けリレーションの入出力と、そのデータ加工（選択、射影、ソートなど）を実行し、結果をマスタプロセッサに返す。PCPはリレーション入力と選択、射影などを並列に処理し、またリレーション入力をファイル管理を経由せずに直接キャッシュマネージャーを介してアクセスし高速化を図るなど、データベース処理の専用ソフトウェアに徹している。

２．５．２　各プログラムの機能

以下に図2.12に対応した各プログラムの機能概要を示す。

［１］ホストプロセッサ上のプログラム

ホストプロセッサ上のプログラムは、ユーザーインタフェースやSQLの解析とその結果の表示等を行う。

①SQLパーサー

SQLを解析し、中間言語に変換する。HDM通信プログラムを介して中間言語をHDMに送り、その結果を受信する。

②SQI

インタラクティブSQL機能を実現するユーザーインタフェースプログラムである。ユーザーからSQL文を受け取り、検索結果を編集して画面に表示する。

③HDM通信プログラム（HDMCOM）

中間言語をHDMに送信し、その結果を受信する。

［２］　HDM制御プログラム

（１）マスタープロセッサ上のプログラム

マスタープロセッサ上のプログラムは、ホストから多数の関係データベース指令を受け取り、それらを同時に実行するとともに、スレーブCPUを制御する。

①ホスト通信プログラム（HCP）

ホストとHDMの通信を司る。ホストから受け取ったデータをトランザクション制御プログラムに渡す。逆にこれから受け取ったデータをホストへ送信する。

②トランザクション制御プログラム（TCP)

HDMはランダムに到着する関係データベース指令を処理しなければならない。そのために多重トランザクション処理が必要である。TCPは、メッセージキューを利用して、多重トランザクションのために重要な役割を果たしている。即ち、HCPから受け取ったトランザクションをチェックした後、メッセージキューの中に入れる。また、実行プログラムが処理した結果を、メッセージキューから取り出し、送信単位に分割してホストプログラムへ渡す。

③トランザクション処理プログラム（TPP)

関係データベース指令を実行するプログラムである。TPPは複数のタスク上で実行され、それらは各々別々のトランザクションを並行に処理する。TPPはメッセージキューからトランザクションを取り出して実行し、その結果をメッセージキューへ出力する。インデックス・スキーマ情報の管理と、その情報による処理はTPPが行うが、リレーションの実際の操作等はスレーブプロセッサ通信プログラム経由で、プロセッサコマンド処理プログラムに要求を出す。例えば、SELECT指令に対しTPPはインデックスの利用等の可否を決定し、スレーブプロセッサ通信プログラムに選択要求を出す。その結果をマージし、メッセージキューに入れる。

④トランザクションディスパッチャ（TDSP)

TPPは各実行タスク上で並列に別々のトランザクションを実行する。TDSPは各トランザクションの並行処理の可否を、トランザクション固有のトランザクションレベルによって決定し、トランザクションの並行実行を制御する。

⑤スレーブプロセッサ通信プログラム（SCP)

SCPは、マスター・スレーブプロセッサの共通メモリ上にあるメールボックスで情報を伝達する。スレーブプロセッサ側は、結果をメールボックスを介してマスタープロセッサに返す。SCPは、スレーブプロセッサの資源管理と同期制御を行っている。結果的に、マスター＝スレーブ間とスレーブ＝スレーブ間の並列制御を行っている。そのためTPPタスクは、他のTPPタスクがスレーブプロセッサを使用しているかどうかということを、気にすることなく要求を出すことが出来る。

⑥キャッシュファイルマネージャー（CFM)

マスタープロセッサは、結合などをのトランザクションの中間結果、または最終結果を貯えておくために大量の作業領域が必要である。キャッシュファイルマネージャーはこのような目的のために作業領域を提供する。大容量のメモリ領域をこのために利用する。もし用意したメモリが無くなれば、ディスク領域上を使用する。このように出来る限りメモリ上で処理することによって高速処理の実現を図る。

（2）スレーブプロセッサ上のプログラム

スレーブプロセッサ上ではリレーションの入力とその加工を行う。以下にスレーププロセッサ上で動作するプログラムについて述べる。

①プロセッサコマンド処理プログラム（PCP）

PCP はマスタープロセッサから要求を受け、リレーションの入出力とそのデータの加工（選択、射影、整列 etc）を実行し、その結果をマスタープロセッサに返す。PCP はスレーブプロセッサ上にあるため、これらはお互いに並列に処理できる（スレーブプロセッサの数だけ）。また、リレーション入力と選択、射影などを、並列に処理し、トランザクション処理時間を短縮することもできる。リレーション入力をファイル管理を経由せずに、直接キャッシュマネージャーを介してアクセスするために、入出力のオーバーヘッドを減らしている。

（3）マスタープロセッサとスレーブプロセッサ上のプログラム

マスタープロセッサと、スレーブプロセッサ両方の CPU で動作するプログラムについて述べる。キャッシュマネージャーとディスクドライバーに関しては後の 2.6 にて記述する。

①ファイルマネージャー（FM）

HDM のファイルは、キードファイルとリレーションファイルで構成する。スキーマ、インデックスは、キードファイル構成である。不要な部分を省略し、オープン、クローズ処理を無くした単純な設計とし、ファイル管理のオーバーヘッドを減らしているのが特徴である。

以上、各プログラムの概略動作仕様について記述した。

２．６　ディスクアクセス方式の特徴

２．６．１　ディスクアクセスの最適化

リレーショナルデータベースでは、リレーションの中からある条件を満たすタプルを捜すのに、インデックスが利用できず、全数サーチ（リレーション内の全タプルの内容を一つ一つ調べる方法）、または部分的全数サーチ（クラスタードインデックスにより限定された範囲内での全数サーチ）を行うケースが意外に多い。通常のリレーショナルデータベースシステムでは、このようにインデックスを利用した標準的なアクセスからちょっと外れた、いわゆる ad hoc な検索を行おうとすると、とたんに膨大な時間がかかってしまうという欠点が存在する。

HDM はこの全数サーチを高速に行えるのが特徴であり、そのために以下のような三つの工夫が行われている。

（１）ディスク上のデータ配置の最適化

（２）大容量のディスクキャッシュメモリ

（３）処理に最適化したディスクアクセス

HDM では、データベース処理プログラムが扱うデータの種類・性質により、ディスク上のデータ格納場所を分離している。そして、その格納場所に対応した専用ディスクキャッシュメモリを経由して、データを高速にアクセスするのが特徴である。

２．６．２　ディスク上のデータ配置

データベース処理プログラムがリレーションからタプルを検索する場合には、まずリレーション名と検索条件からリレーションのディレクトリ情報を調べ、これに基づいて条件を満たすタプルをリレーションの中から探索する。この時全数サーチ処理が必要な場合には、リレーションのタプルは大量に連続的にアクセスされる。それに比べるとディレクトリ情報は少量しかアクセスされない。このため全数サーチを行うデータの部分は、ある程度大きな単位でリレーションの入出力を行う方が効率がよく、ディレクトリ情報の部分は細分化してダイレクトアクセス性を高めるのが効率が良い。

このために HDM では、ディスクをディレクトリ領域とリレーション領域の二つに大き

く分けている。ファイルディレクトリ、リレーションディレクトリ、インデックスファイル等のデータはディレクトリ領域へ格納し、リレーションの実際のタプルはリレーション領域へ格納する。ディレクトリ領域とリレーション領域とでアクセス方法を専用化して、ディレクトリ領域はトラック単位(18KB)、リレーション領域は論理シリンダ単位（半シリンダ、90KB）にデータの入出力を行うようにしている。

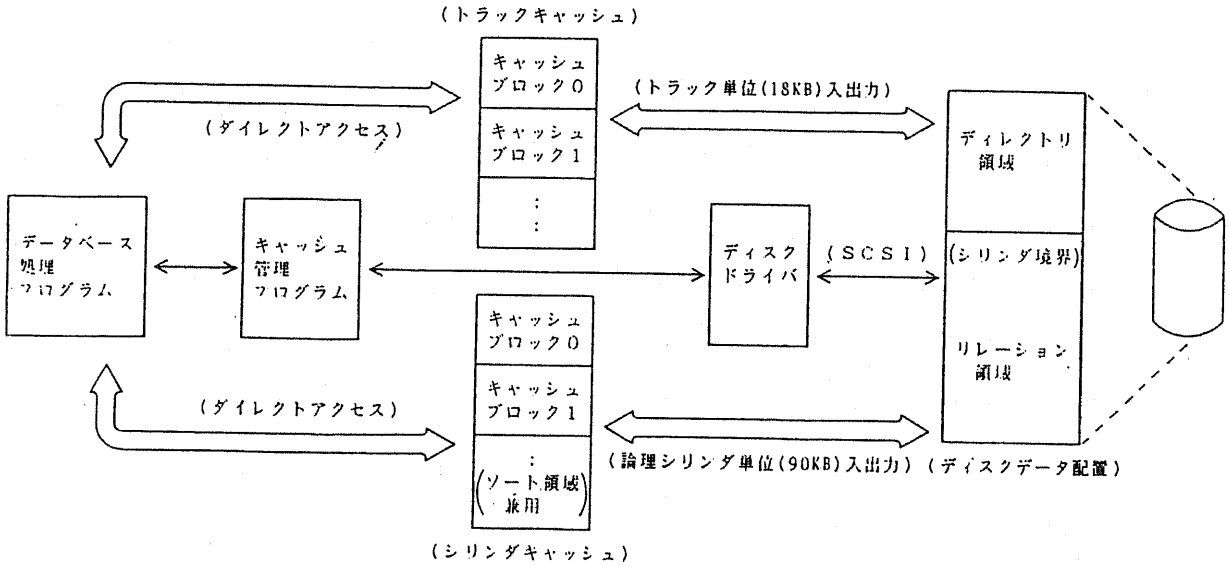


図 2.13　ディスクアクセス処理の流れ

２．６．３　大容量ディスクキャッシュメモリ

HDM では、高速に効率よくディスクアクセスを行う目的で、主記憶上に大容量のディスクキャッシュ領域を持っている（3MB/1 プロセッサ）。 ディスクキャッシュにはトラックキャッシュとシリンダキャッシュとがあり、それぞれディスク上のディレクトリ領域とリレーション領域とに対応する。（図 2.13）それぞれの領域はシリンダ境界で分離してあるので、ディレクトリとリレーションとが同一キャッシュ内に共存することはない。

一般のディスクキャッシュでは、キャッシュのデータは一旦主記憶上の別の領域に移され、それに対してプログラムがアクセスする。これに対し HDM では多くの場合、データベース処理プログラムが直接ディスクキャッシュ上のデータをアクセスする。これはデータベース専用マシンなるが故に可能な技であるが、これにより主記憶上のデータコピー回

数を大幅に減らすことが出来、処理の高速化が図られる。

当然のことながら、データが既にディスクキャッシュ上に存在する場合には、データをディスクから入力せずにディスクキャッシュ上のデータが利用できる。これは同一リレーションを頻繁に参照する場合とか、トランザクションを実行する毎に必ず参照される中核的ディレクトリ情報等へのアクセスに対して有効である。

２．６．４　ＲＤＢ処理に専用化したソフトウェア

HDM では複数のプログラムがディスクキャッシュを共有して効率良くアクセス可能にするために、ディスクキャッシュ管理プログラムを実装しているが、以下にその概要を示す。

・データベース処理プログラムは処理したいデータがあると、トラックキャッシュかシリンダキャッシュかを区別し、ディスクアドレスを指定してディスクキャッシュ管理プログラムに要求を出す。

・ディスクキャッシュ管理プログラムはこの要求に基づいて、ディスクアドレスで指定されるトラック又はシリンダを、空いているキャッシュブロックへ読み込む。次に、指定されたディスクアドレスに対応する主記憶上のアドレスをデータ処理プログラムへ返す。

・データベース処理プログラムは、この主記憶上のアドレスをもとにして、直接ディスクキャッシュをアクセスする。このキャッシュブロックは、データベース処理プログラムが、キャッシュ管理プログラム経由で解放するまでは、主記憶上で有効となる。

その他のプログラムも、基本的にこのディスクアクセス方式をベースにして作成されている。以下にこれらの例を示す。

・ディスクドライバもこのディスクアクセス処理に適合するように作られており、それ以外の汎用的処理部分のロジックは持っていない。このためロジックは単純化されメンテナンスも容易になっている。

・リレーションの中から全数サーチでタプルを検索する場合には、データベース処理プログラムは論理シリンダ単位にダブルバッファリングを行いデータを読み出す。このダブルバッファリングにより、CPU が実行する条件を比較して目的のタプルを捜す処理と、ディ

スクからのデータ入出力と並行して実行出来るため、全件サーチ処理が高速に行える。

・また処理の途中で、選び出したタプルのソート処理が必要となる場合には、一時的に主記憶上のシリンダキャッシュ領域をつぶし、ここにソート領域をダイナミックに割り付けるということを行う。これにより、大量の主記憶を使って高速にソート処理を行うことが出来る。

以上のように、リレーショナルデータベース処理に最適化したディスクアクセスロジックを構成することにより、処理の高速化を図っている。

２．７　高信頼化機能

本節ではHDMで実現している高信頼化機能として、データベースの障害回復機能について述べる。

２．７．１ データベースシステムの障害回復

一般にデータベースシステムの障害回復方法としては、UNDO／REDOログ、チェックポイント、およびダンプ／ロードといった技法が知られている［Gray-78］。更新処理の途中でシステムがダウンした場合、データベースの首尾一貫性を保つため、その処理によって更新されたデータを元に戻すのに使用されるのがUNDOログであり、また、システム再立ち上げ時に、障害発生前に正常終了した更新処理を再実行するのに用いられるのがREDOログである。一般にUNDO／REDOログは、それぞれ更新前、及び更新後の値をディスクなどに記録したものである。チェックポイントはREDO、およびUNDOの処理量（ログ量）を減らすために、バッファ中の更新された値を定期的にディスクに書き出す技法である。また、ダンプ／ロードはディスククラッシュなどの障害からの回復を行うための技法で予め他の媒体（MTなど）にコピー（ダンプ）しておいた全データを、障害発生時にディスクにロードすることによって回復処理を行うものである。ロード後にREDOをおこなうことにより、障害発生直前の状態まで回復することが出来る。

HDMは、これらの技法（UNDO／REDOログ、チェックポイント、ダンプ／ロード）を、以下に述べる手法で実現している。以下、HDMにおける各技法の実現方式、および特徴について述べる。

２．７．２ ＵＮＤＯログ

HDMは、１台のマスタープロセッサと複数台のスレーブプロセッサからなるマルチプリセッサのデータベースマシンであり、それらの並列処理と、各プロセッサに搭載した大容量メモリのキャッシュとしての使用により高速処理を実現している。これらの点を考慮して、HDMでは以下のようにしてUNDOログ機能を実現する。

・各プロセッサで個別にログを取得する。

・トランザクション単位に、更新前の物理ページイメージとして、ディスク上のUNDO

ログ領域に取得する。

・ログ取得効率を向上させるため、各プロセッサのメモリ上に設けたUNDOログバッファによりバッファリングを行う（図2.14参照）。あるページが、ログバッファからディスク上のログ領域へ書き出されるのは、以下の２つの場合である。

①ログバッファ満杯時

②キャッシュ上の対応するページがディスクに書き出される直前

従って、更新されたページがキャッシュ上にあるうちはそのページのログを書き出す必要はないためUNDOログの量を減らすことが出来る。

・UNDOは、ログバッファ、及びログ領域から読み込んだ更新前イメージを逆順に（すなわち新しいものから順に）、該当するページに物理的に重ね書きすることによって行う。

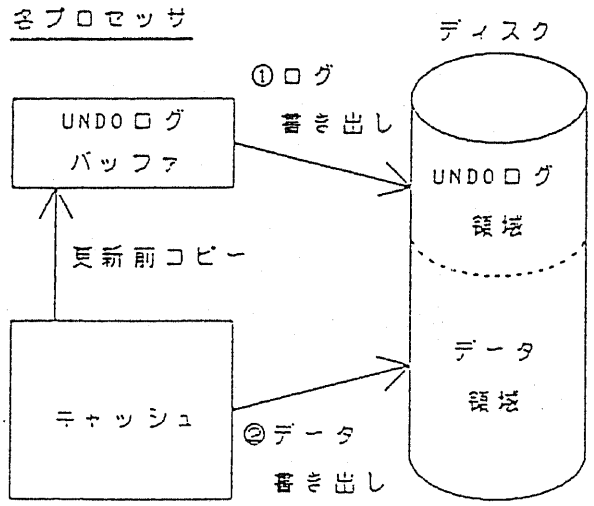


図2.14　UNDOログの管理方法

２．７．３ ＲＥＤＯログ

従来の障害回復技法では、REDOログとして、更新後の物理ページイメージ、あるいは更新後のレコードの値そのものを用いていた。しかしリレーショナルデータベースでは、１つのＳＱＬ文で大量データの更新が簡単に行えてしまうため、従来方式ではログの量が膨大なものになってしまうという問題点があった。

そこでHDMでは、更新後の値をREDOログとして記録しておかなくても、どのような

更新処理を行ったかさえ記録しておけば、その処理を再実行することにより回復を行うことができるという点に着目し、REDO ログとして、更新処理の SQL 文（実際は、その HDM の中間言語による表現）を取得することとした。これにより、回復に要する時間は少し長くなるものの、REDO ログの量を大幅に減らすことが出来る。以下に具体的な処理方法を示す。

① REDO ログは、マスタープロセッサのディスク上に設けた REDO ログ領域に、HDM の中間言語形式で取る。中間言語には、トランザクション ID や実行したユーザの ID など、再実行に必要な情報が記されている。

②トランザクション管理プログラムは、１つの更新処理単位（action、すなわち、1 つの SQL 文）が正常終了した時、その中間言語をログ領域に書き出し、ユーザに処理結果を返す。

③ログ領域が満杯近くなったとき（８０%を越えたとき）、フロントエンドの計算機に接続されたカートリッジ MT などに REDO ログをコピーする（HDM は図 2.15 に示すように、フロントエンドの計算機として UNIX ワークステーション（ME シリーズ）に SCSI で接続するようになっている）。

④ REDO は、コミットされたトランザクションの各 action を時間順に再実行することによって行う。

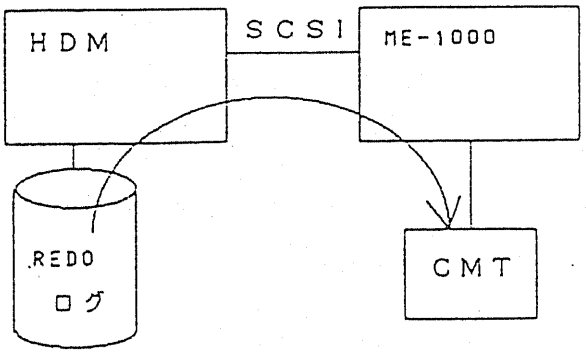


図 2.15　REDO ログのコピー

２．７．４ チェックポイント

HDM では、UNDO ログの量を減らすためと、システム回復時間を短縮するために、チェックポイントを設けている。チェックポイントは、Transaction Consistent Checkpoint

とする。すなわち、トランザクションの切れ目でチェックポイントを取る。またチェックポイントでは、キャッシュがディスクに書き出されるため、ディスク上のデータベースは首尾一貫した状態になる(UNDO が不要になる)。そのため、チェックポイントでの UNDO ログの消去が可能になる。チェックポイントは、以下の場合に取られる。

・任意のプロセッサのディスク上のログ領域が満杯に近くなったとき（８０%を越えたとき）。

・前回のチェックポイントから一定時間（例えば、５分）経過したとき。

また、チェックポイントでは以下の処理を行う。

・キャッシュのディスクへの書き出し

・UNDO ログの消去

・REDO ログ領域が満杯に近くなっている場合は、フロントエンドの CMT 当への REDO ログのコピー

２．７．５ ダンプ／ロード

ダンプは、各テーブル、インデックス、およびレコードを生成する SQL 文の中間言語を、フロントエンドの CMT（カセット磁気テープ装置）等にセーブすることによって行う。ロードは、セーブされたダンプファイル（中間言語）をそのまま再実行することによって行われる。この論理ダンプ／ロードによって、ディスクの特定部分が破壊された場合でも、データを正しくロードでき、また、特定のテーブルのみのダンプ／ロードも容易に行うことが出る。なお、SQL の insert 文では、同時に１つのレコードしか追加できず効果が悪いため、複数のタプルを一括してストアできる multi-insert 機能を設けた。これにより、ロードに要する時間を短縮することが出来る。

２．７．６ まとめ

以上をまとめると HDM の障害回復方式は、以下のような長所を持っている。

・更新されたページをキャッシュからディスクに書き出す直前に UNDO ログをディスクに書き出すため、更新されたデータがキャッシュ上にあるうちは UNDO ログを書き出さなくてもよく、ログ量を減らすことができる。

・REDO ログを、SQL 文で取得することにより、ログ量を大幅に減らすことが出来る。
・チェックポイント時に UNDO ログを消去できるため、ログ量を減らすことができる。
・データを SQL 文に変換してダンプ（論理ダンプ）するため、ディスク上の特定の（物理的な）位置が破壊されて使用できなくなった場合でも、データを正しくロードできる。

以上のように HDM では、障害回復の考え方を単純化し、新しい工夫を盛り込んで、効率の良い障害回復機能を実現している。

２．８　性能評価

本節では、最初に、研究所における試作機の性能評価の結果について述べ、最後に商用版の性能評価結果について述べる。

２．８．１　測定環境

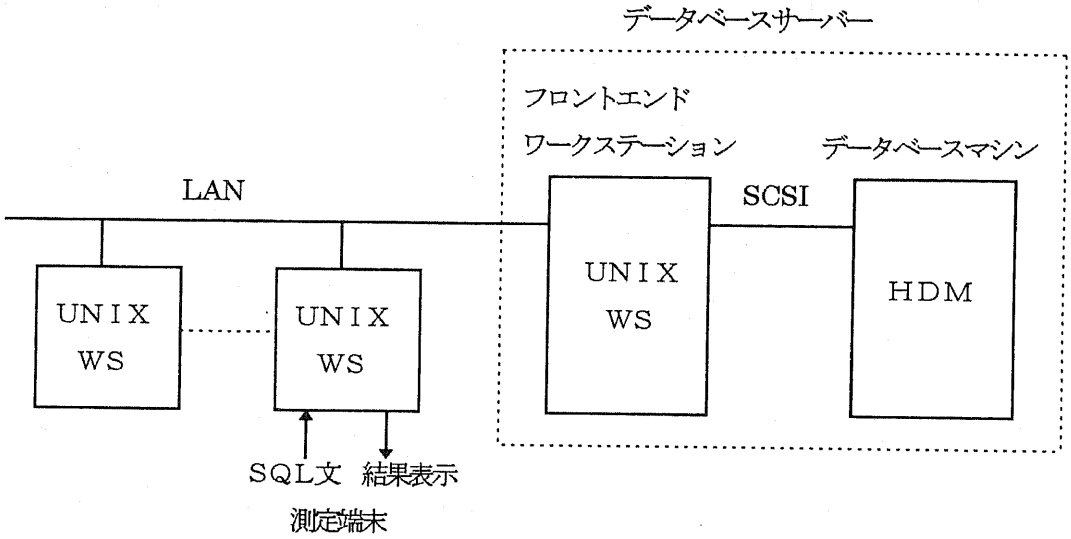


図 2.16　性能評価時のシステム構成

HDM の性能評価は図 2.16 のシステム構成にて行った。即ち、HDM とフロントエンドマシン（UNIX ワークステーション）を SCSI で結合してデーターベースサーバーを構成し、これに対し LAN 上のワークステーションから SQL 文による問い合わせを行い、結果が表示されるまでの経過時間を測定する。HDM では検索結果のすべてが得られたあとにユーザー画面に結果が表示される。即ち、例えば検索結果が 1000 タプルである時、最初の 10 タプル程度の検索結果が得られた時点で画面表示を開始するようなことがない。このため端末での測定により全検索時間が正確に測定できる。

又 HDM の測定は HDM 上のディスクキャッシュをクリアーした後の１回目の測定結果と、引き続き行った２回目の測定結果について並記してある。これは使用したベンチマークにディスクキャッシュに関する規定がないため、参考としてディスクキャッシュの効果を明確に示すため 1st と 2nd の２つを並記したものである。又これら全体に対し２回の測定を行いその平均値を結果として示した。

２．８．２　ウィスコンシンベンチマークによる性能評価

Wisconsin Benchmark［Bitton-83］は、RDB の標準的なベンチマークとして極めて有名なものの１つである。Wisconsin Benchmark では 1000 タプルのリレーション（t1k）と 10000 タプルのリレーション（t10k）を使用する。各タプルは 13 項目（アトリビュート）の２バイト整数、３項目の 52 バイト長文字列からなる。各項目の値は乱数によって決定されたものである。表 2.1～2.6 に HDM における端末上の応答時間を計測した結果を示す。測定は、全ディスクキャッシュを無効にした直後の１回目の処理と２回目の処理の２つの場合について行った。１回目の処理では全データをディスクから読み込む必要があるため、その分より多くの処理時間を要する。比較のため、他のデーターベース・システムの結果も併せて示す。INGRES、ORACLE、IDM の性能値についてはそれぞれ［Bitton-83］，［Oracle］，［Simon-85］から引用した。これらの測定の動作環境は次の通りである。

INGRES　：　software DBMS on VAX11/750

ORACLE　：　software DBMS on VAX11/750

IDMnodac：　database machine IDM500(without DAC)

IDMdac　：　database machine IDM500(with DAC)

(DAC:database accelerator)

（１）選択

次の２つの問い合わせによって、選択演算の性能評価を行った。

(a) 選択率 1%の選択演算

insert into temp select * from t10k where uniquel between 100 and 199 ;

(b) 選択率 10%の選択演算

insert into temp select * from t10k where uniquel between 1000 and 1999 ;

ここで、t10k は１万タプルのリレーション、uniquel は 0-9999 の一意の値を持つ２バイトの整数項目である。これらの検索をインデックスがない場合と、ある場合の両方について測定した。表 2.1 がインデックスがない場合の性能評価結果である。上記（a）の問い合わせの結果が表中の"100"の項目に、上記（b）の問い合わせの結果が"1000"の項目に示される。表の中の数字は問い合せの経過時間（秒）であり、この数字が小さいほど性能がよいことになる。この結果から HDM は他のシステムに比べ 10 倍以上の高性能を達成している

ことがわかる。表 2.2 は同様にインデックスがある場合について上記（a），(b) の問い合わせを行った結果である。インデックスがクラスタードインデックスの場合と非クラスタードインデックスの場合のそれぞれについて測定した。一般にインデックスが利用できる場合には HDM の並列処理が生かされていくため、他のシステムとの差は現れにくい。しかし非クラスタードインデックス付きで 1000 件といった多数のタプルを抽出する場合には、選択した後のタプル抽出時に HDM の並列処理が効果を発揮するため、HDM は他のシステムに比べ 10〜20 倍高性能な結果となっている。

表 2.1 Selection Queries without Indices
(Result Tuples Inserted into Relation)

[Total Elapsed Time in Seconds]

System	Number of Tuples Selected from 10,000 Tuple Relation	
	100	1000
INGRES	38.40	53.90
ORACLE	53.20	72.50
IDM(nodac)	20.30	27.20
IDM(dac)	19.90	23.40
HDM(1st)	1.89	2.03
HDM(2nd)	1.25	1.41

表 2.2 Selection Queries with Indices
(Result Tuples Inserted into Relation)

[Total Elapsed Time in Seconds]

System	Number of Tuples Selected from 10,000 Tuple Relation			
	Clustered Index		Non-Clustered Index	
	100	1000	100	1000
INGRES	3.90	18.90	11.40	54.30
ORACLE	4.50	31.60	7.30	53.50
IDM(nodac)	1.80	8.90	4.10	29.80
IDM(dac)	1.10	7.40	3.30	24.80
HDM(1st)	1.75	2.46	2.17	2.81
HDM(2nd)	1.11	1.43	1.15	1.86

（2）結合

以下の 3 つの問い合わせによって、結合演算の性能評価を行った。この結果を表 2.3 に示す。

(a) insert into temp select t1.*, t2.* from t10k t1, t10ka t2

where t1.unique1 = t2.unique1 and t2.unique1 < 1000 ;

(b) insert into temp elect t1.*, t2.* from t10k t1, t10ka t2
where t1.uniquel =t2.uniquel ;

(c) insert into temp select t1.*, t2.* from t1k t1, t10k t2, t10ka t3
where t1.uniquel = t2.uniquel and t2.uniquel = t3.uniquel
and t2.uniquel < 1000 and t3.uniquel < 1000 ;

表 2.3 Join Queries

[Total Elapsed Time in Seconds]

System	Query		
	joinAselB	joinABprime	joinCselAselB
	Without Indices		
INGRES	108.00	156.00	126.00
ORACLE	>18000	>18000	>18000
IDM(nodac)	570.00	552.00	126.00
IDM(dac)	80.00	114.00	36.00
HDM(1st)	6.12	5.84	7.36
HDM(2nd)	5.39	5.25	6.59
	With Clustered Indices		
INGRES	54.00	102.60	64.20
ORACLE	77.50	73.50	111.40
IDM(nodac)	28.20	25.80	40.80
IDM(dac)	22.20	14.40	28.20
HDM(1st)	5.95	5.77	7.18
HDM(2nd)	4.92	4.99	6.03
	With Non-Clustered Indices		
INGRES	118.20	108.00	114.60
ORACLE	139.90	87.30	172.10
IDM(nodac)	52.80	40.20	75.00
IDM(dac)	39.00	37.80	48.60
HDM(1st)	6.02	5.72	7.25
HDM(2nd)	5.30	5.14	6.50

表 2.3 の結果から、HDM が他のシステムに比べ Join 性能がきわめて高いことが分かる。インデックスがない場合にはその差が特に顕著である。

（3）射影

次の２つの問い合わせによって射影演算の性能評価を行った。

(a) insert into temp select distinct hundred from t10k ;

(b) insert into temp select distinct * hundred from t1k ;

ここで、hundred は、0-99 の値を持つ 2 バイトの整数項目であり、t1k は 1000 タプルのリレーションである。 (a) の問い合わせでは、1 万件のタプルがソートされ、重複除去され

てタプル数が 100 件のテンポラリリレーションが生成される。また（b）の問い合わせでは、実際に除去されるタプルはないが、それでも重複除去のため 1000 タプルのソート処理は行われる。

表 2.4　Projection Queries
(Duplicate Tuples are Removed)

[Total Elapsed Time in Seconds]

System	100/10,000	1000/1000
INGRES	26.40	132.00
ORACLE	117.30	29.50
IDMnodac	58.90	31.50
IDMdac	33.00	22.00
HDM(1st)	2.56	2.07
HDM(2nd)	1.85	1.84

この射影演算の測定結果を表 2.4 に示す。表 2.4 から明らかなように HDM は他のシステムに比べて、きわめて高い性能（10～60 倍）を達成している。

（4）集約演算

以下の 3 つの問い合わせによって、集約演算の性能評価を行った。

(a) 2 バイト整数項目の最小値を求める演算

insert into temp select min(uniquel) from t10k ;

(b) 2 バイト背数項目の値によって全体を 100 個の部分集合に分割し、各部分集合に対して、他の 2 バイト整数項目の最小値を求める演算

insert into temp select min(twothous) from t10k group by hundred ;

(c) 2 バイト背数項目の値によって全体を 100 個の部分集合に分割し、各部分集合に対して、他の 2 バイト整数項目の合計値を求める演算

insert into temp select sum(twothous) from t10k group by hundred ;

ここで、uniquel、twothous、hundred は、それぞれ 0-9999、0-1999、0-99 の値を持つ 2 バイト整数項目、t10k は 1 万タプルのリレーションである。測定結果を表 2.5 に示す。インデックスがない場合、HDM は他のシステムの 20～700 倍以上の高速処理を実現している。最小値（演算（a））はインデックスを利用することによってより高速に求めることが出来る（IDM、ORACLE で実現）が、HDM では未実装である。

表 2.5　Aggregate Queries

[Total Elapsed Time in Seconds]

System	Query Type		
	MIN Scalar Aggregate	MIN Aggregate (100 Partitions)	SUM Aggregate (100 Partitions)
	Without Indices		
INGRES	34.00	495.00	484.80
ORACLE	62.90	1198.10	1219.80
IDMnodac	22.00	61.50	62.30
IDMdac	19.00	36.00	36.00
HDM(1st)	1.68	2.49	2.55
HDM(2nd)	0.84	1.89	1.87
	With Indices		
INGRES	37.20	242.20	254.00
ORACLE	1.40	1200.10	1211.80
IDMnodac	22.00	61.50	62.30
IDMdac	19.00	36.00	36.00
HDM(1st)	1.82	2.63	2.69
HDM(2nd)	0.90	1.93	1.92

（5）追加・削除・更新演算

以下の 3 つの問い合わせによって、追加、削除、更新演算の性能評価を行った。

・追加（Append）

insert into t10k values (10001, 74, 0, 2, 0, 10, 50, 688, 1950, 4950, 9950, 1, 0,

"mx･･･xgx･･･xc", "gx･･･xcx･･･xa", "0x･･･x0x･･･x0") ;

・削除（Delete）

delete from t10k where unique1=10001 ;

・更新（Modify）

update t10k set unique1 = 10001 where unique1 = 1491 ;

これらの測定結果を表 2.6 に示す。HDM は単に 1 タプルを追加するだけの追加演算に関しては他のシステムとほぼ同程度の処理時間であるが、削除・更新演算に関しては、他のシステムの 20 倍以上の高速処理を実現している。削除・更新演算ではリレーション内のすべてのタプルを検索して該当するタプルを見つけだす必要があるのに対し、追加演算では全件検索を行う必要がない。このような結果が得られたのは、HDM が他のシステムに比べて全件検索を高速に行うことが出来るからである。

表 2.6　Update Queries

[Total Elapsed Time in Seconds]

System	Query Type			
	Append	Delete	Modify	
	Without Indices			
INGRES	1.40	32.30		32.80
ORACLE	1.00	46.70		32.00
IDMnodac	0.90	22.80		29.50
IDMdac	0.70	20.80		20.90
HDM(1st)	1.31	1.45		1.25
HDM(2nd)	0.60	0.78		0.74
	With Indices			
			Cluster	Non-Cluster
INGRES	2.10	0.50	1.60	1.60
ORACLE	1.20	0.90	0.90	0.90
IDMnodac	0.90	0.40	0.60	0.50
IDMdac	0.80	0.40	0.50	0.50
HDM(1st)	1.07	1.04	—	1.02
HDM(2nd)	0.56	0.69	—	0.66

2．8．3　拡張ウィスコンシンベンベンチマークによる性能評価

拡張ウィスコンシンベンチマーク [DeWitt-87] は、ウィスコンシンベンチマークを、より大きなリレーション（テーブル）が扱えるように機能拡張したものである。即ち数千～100 万タプルの規模のリレーションが扱えるようになっている。この中のすべてのタプルは 208 バイト長で、13 個の 4 バイト・インテジャー・アトリビュート（項目）と、3 個の 52 バイト・文字列アトリビュートから成っている。性能測定環境は、2.8.2 と全く同様である。以下に出てくる性能測定結果の表には、HDM と比較する他のシステムとして、Teradata と GAMMA が出てくるが、これは次のようなデータベースマシンである。

・Teradata

商用データベースマシン テラデータ DBC/1012

並列度＝24　（80286 を使ったプロセッサを 24 個使用した版。（インターフェースプロセッサ 4 個、アクセスモジュールプロセッサ(AMP)20 個））

・GAMMA

ウィスコンシン大学で開発されたデータベースマシンの試作機。

並列度＝17　（17 台の VAX11/750 をリングバスにつないで構成。）

ちなみに HDM は、

・HDM

並列度＝5　(68020 を使ったプロセッサを 5 個使用。（マスタープロセッサ 1 個、スレーブプロセッサ 4 個)

なお、Teradata とか GAMMA は非常に高価なデータベースマシンであるのに対し、HDM は安価なデータベースマシンである点も特記すべき事項である。 ちなみに Teradata とか GAMMA は数億円するが、HDM は数百万円で買えるものである。Teradata と GAMMA の性能値については［DeWitt-87］から引用した。

（1）選択

表 2.7 に選択演算の性能評価結果を示す。選択演算としてはそれぞれ 1 万タプル、100 万タプルのリレーションから選択率 1%ないし 10%の選択演算を実行した。表中 (a)，(b) がインデックス無しの時の測定結果、(c)，(d) が非クラスタードインデックスが付いているときの測定結果、(e)，(f)，(g) がクラスタードインデックスが付いているときの測定結果である。 (a) ～ (f) では検索結果は、新しいリレーションが作られてその中に挿入される。 (g) では検索結果は画面に表示される。

表 2.7　Selection Queries

(a)　1% nonindexed selection
(b)　10% nonindexed selection
(c)　1% selection using non-clustered index
(d)　10% selection using non-clustered index
(e)　1% selection using clustered index
(f)　10% selection using clustered index
(g)　1 tuple selection using clustered index

[Total Elapsed Time in Seconds]

Source Relation	Database Machine	(a)	(b)	(c)	(d)	(e)	(f)	(g)
10,000 tuples	Teradata	6.86	15.97	7.81	16.82	—	—	—
	Gamma	1.63	2.11	1.03	2.61	0.59	1.26	0.15
	HDM(1st)	2.02	2.09	2.27	2.99	2.00	2.49	0.96
	HDM(2nd)	1.18	1.40	1.21	1.89	1.13	1.41	0.61
100,000 tuples	Teradata	28.22	110.96	29.94	111.40	—	—	1.08
	Gamma	13.83	17.44	5.32	17.65	1.25	7.27	0.15
	HDM(1st)	6.77	8.08	8.89	16.05	4.66	11.25	0.97
	HDM(2nd)	6.55	8.41	8.53	16.27	2.45	11.53	0.58
1,000,000 tuples	Teradata	213.13	1106.86	222.65	1107.59	—	—	—
	Gamma	134.86	181.72	53.86	182.00	7.50	69.60	0.20
	HDM(1st)	81.94	126.38	71.66	184.98	30.47	137.33	0.91
	HDM(2nd)	82.83	131.68	71.85	187.53	30.70	139.44	0.54

表 2.7 から、HDM はすべての場合について Teradata よりも早く、又インデックスがない場合には GAMMA よりも速いことがわかる。

（2）結合

結合演算の性能評価に使われる問い合わせは、リレーションのサイズが異なることを除いて、2.8.2 のウィスコンシンベンチマークの時と同じである。

表 2.8 Join Queries

(a) joinAselB (without indices)
(b) joinABprime (without indices)
(c) joinCselAselB (without indices)
(d) joinAselB (with indices)
(e) joinABprime (with indices)
(f) joinCselAselB (with indices)

[Total Elapsed Time in Seconds]

Source Relation	Database Machine	(a)	(b)	(c)	(d)	(e)	(f)
10,000 tuples	Teradata	35.60	34.90	27.80	25.00	22.20	23.80
	Gamma	5.10	6.50	7.00	5.00	5.70	7.20
	HDM(1st)	6.36	6.03	7.71	6.21	5.91	7.43
	HDM(2nd)	5.55	5.43	6.72	5.16	5.15	6.17
100,000 tuples	Teradata	331.70	321.80	191.80	170.30	131.30	156.70
	Gamma	36.30	46.50	38.40	36.90	45.60	37.90
	HDM(1st)	69.03	62.36	79.45	70.44	63.32	80.91
	HDM(2nd)	70.08	63.21	79.88	71.10	64.14	81.30
1,000,000 tuples	Teradata	3534.50	3419.40	2032.70	1584.30	1265.10	1509.60
	Gamma	703.10	2938.20	731.20	737.70	2926.70	712.80
	HDM(1st)	1123.79	1028.71	1233.16	1019.26	968.58	1118.39
	HDM(2nd)	1128.69	1031.94	1237.59	1022.00	970.17	1120.27

表 2.8 に性能測定結果を示す。この表に示されるように、HDM は全てのケースについて Teradata よりも速く、多くのケースについて、GAMMA よりも速い。

（3）集約演算

集約演算についても (2) と同様、リレーションサイズが大きくなったこと以外は、2.8.2 のウィスコンシンベンチマークの時と同様である。この性能評価結果を表 2.9 に示す。
表 2.9 から、HDM はほとんどのケースにおいて、他のシステムよりも高速であることが分かる。

表 2.9 Aggregate Queries

[Total Elapsed Time in Seconds]

Source Relation	Database Machine	MIN scalar aggregate	MIN aggregate (100 partitions)	SUM aggregate (100 partitions)
10,000 tuples	Teradata	4.41	8.66	8.94
	Gamma	1.89	2.86	2.89
	HDM(1st)	1.80	2.61	2.61
	HDM(2nd)	0.88	1.91	1.91
100,000 tuples	Teradata	18.29	27.06	24.79
	Gamma	15.53	19.43	19.54
	HDM(1st)	6.18	14.86	14.97
	HDM(2nd)	5.96	14.65	14.77
1,000,000 tuples	Teradata	127.86	175.95	175.78
	Gamma	151.10	184.92	185.05
	HDM(1st)	80.81	176.35	177.33
	HDM(2nd)	80.59	176.21	177.16

（4）更新演算

追加演算（(a)，(b)）、削除演算（(c)，(d)）、更新演算（(e)，(f)）を、それぞれ 1 万タプル、10 万タプル、100 万タプルのリレーションに対して、実施したときの性能値を表 2.10 に示す。この時の HDM では、クラスタインデックスに指定されたアトリビュート（項目）に対する更新は行えなかったため（変わりに削除&追加の 2 段階で実行する）、表 2.10 中の (d) の HDM のところは空白になっている。表 2.10 のその場所を除いて、HDM は概ね他のシステムより高速であることが分かる。

表 2.10 Update Queries

(a) Append 1 tuple (no indices exist)
(b) Append 1 tuple (one index exists)
(c) Delete 1 tuple using the key attribute
(d) Modify 1 tuple using the key attribute (the key attribute is modified)
(e) Modify 1 tuple using the key attribute (a non-indexed attribute is modified)
(f) Modify 1 tuple using a non-key attribute with non-clustered index (the non-key attribute is modified)

[Total Elapsed Time in Seconds]

Source Relation	Database Machine	(a)	(b)	(c)	(d)	(e)	(f)
10,000 tuples	Teradata	0.87	0.94	0.71	2.62	0.49	0.84
	Gamma	0.18	0.60	0.44	1.01	0.36	0.50
	HDM(1st)	0.77	0.89	0.99	—	0.94	0.91
	HDM(2nd)	0.63	0.50	0.61	—	0.61	0.60
100,000 tuples	Teradata	1.29	1.62	0.42	2.99	0.90	1.16
	Gamma	0.18	0.63	0.56	0.86	0.36	0.46
	HDM(1st)	0.70	0.93	1.03	—	0.92	0.96
	HDM(2nd)	0.61	0.50	0.60	—	0.65	0.53
1,000,000 tuples	Teradata	1.47	1.73	0.71	4.82	1.12	3.72
	Gamma	0.20	0.66	0.61	1.13	0.36	0.52
	HDM(1st)	0.71	0.86	0.90	—	0.75	0.80
	HDM(2nd)	0.63	0.49	0.59	—	0.58	0.53

２．８．４　ＨＤＭ商用版の性能評価

“三菱高速データ検索システム”の機種名で商用化された版の性能評価結果について述べる。これに先立ち試作機から商用版に向けて、性能強化が図られたのでその概要を示す。

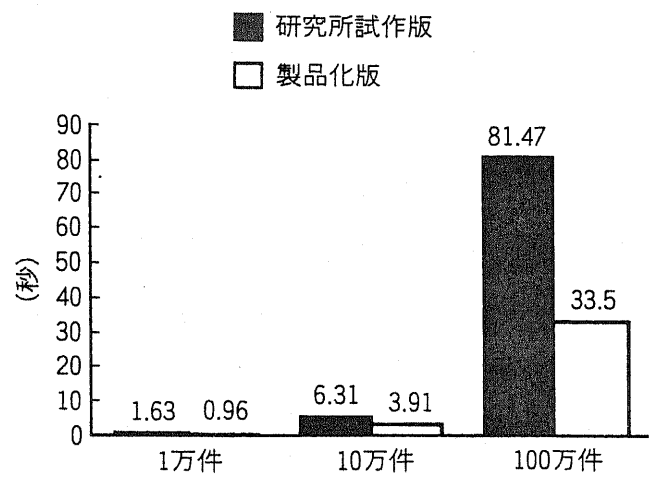
（１）商用版に向けた性能強化の概要

プロセッサの処理速度を上げるため、システムロックを 16MHz から 20MHz に変更した。また、メモリアクセス回路を高速化し 10M バイト／秒から 13.3M バイト／秒とした。さらに、SCSI バスの転送方式を非同期転送から同期転送に変更することによってデータ転送レートを 2M バイト／秒から 5M バイト／秒に引き上げた。

これらの改良により、製品版では試作機の 1.6 倍から 2.4 倍の処理速度を実現した。

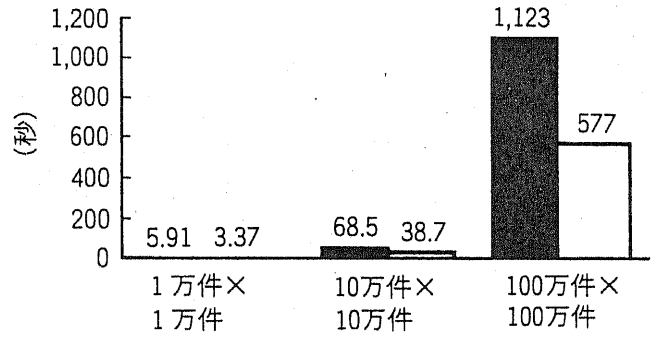
（２）商用版の性能向上

試作機と商用版の性能比較のため、図 2.17 にそれぞれの拡張ウィスコンシンベンチマークテストの測定値を示す。リレーション（表）は、１タプル 208 バイトの、１万件、10 万件、100 万件のものを使用している。初回検索のキャッシュ無効状態で“選択”“結合”“集約”について高速データ検索マシンと試作機の内部実行性能を比較した。この図から高速データ検索マシンが試作機に比べて 1.6 倍から 2.4 倍性能向上していることが分かる。



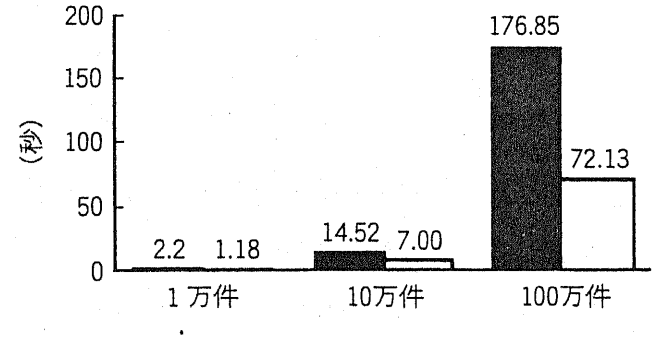
(a) 選択（選択率１％の選択演算）

```
insert into templ select * from flok
where unique2 between 1000 and 1099 ;
```



(b) 結合（２つのテーブルの10%選択条件付き結合）

```
insert into templ select f 1 *f 2 *from
folk f 1 , floknl f 2 where f 1 . unique 2 =f 2 . unique 2
and f 2 . unique 2 <1000 ;
```



(c) 集約（100分割した部分ごとの
４バイト整数項目の合計値）

```
insert into templ select sum(twothous)
from flok group by hunded ;
```

図 2.17 拡張ウィスコンシンベンチマークテスト比較結果

（3） 商用版と他機種の性能比較

三菱高速データ検索システムの検索性能を、クライアント／サーバ型データベースである SQL Base と、ファイルサーバ型データベースである dBase III とで比較した（表 2.11）。表中の値は画面に検索条件を設定後、命令を実行させてから結果が画面に出力されるまでの時間を示している。検索対象としては、1 タプル 500 バイト、50 列のリレーションを用いている。

表 2.11　検索性能比較（入力件数特性）

		1 万件	5 万件	10万件	50万件
ファイル サーバ型	dBaseIII	40秒	202秒	—	—
クライアント ／サーバ型	SQL Base	47秒	235秒	—	—
	高速データ検索システム	1 秒以下	5 秒	8 秒	38秒

このシステムに対しては 1 万件から 50 万件の大規模なテーブルを対象としたが、SQL Base, dBase III の場合は現実的な処理時間を考慮して 1 万件から 5 万件を対象とした。5 万件での結果を比較すると、このシステム（HDM）が他のデータベースシステムよりも 40 倍以上高速な性能を達成していることがわかる。

２．８．５ 性能評価のまとめ

2.8.2 では、1 万タプル（レコード）規模の比較的小規模のリレーション（テーブル）を用いて、HDM の性能測定を行い、同程度のシステム規模の他のデータベースシステムとの性能比較を行った。この結果原理的に差が出にくい追加（Append）とか、インデックスが利用できる時の検索（select）等の簡単な場合を除いて、HDM は、他のシステムより数 10 倍高速であることが分かった。特に典型的なリレーショナルデータベース演算である。結合（Join）、射影（Projection）、集約（Aggregate）といった重負荷な処理で HDM は真価を発揮し、場合によっては 100 倍以上の高性能を達している。

2.8.3 では 1 万～100 万タプルの大規模リレーションを用いて性能評価を行った。ここで

はテラデータ DBC/1012 という 24 プロセッサの商用データベースマシン、ウィスコンシン大学で試作された GAMMA というミニコン（VAXII-750）を 17 台つなげたデータベースマシン（いずれも数億円規模のシステム）と、HDM（5 プロセッサ、数百万円規模）の性能を比較した。この結果、HDM はテラデータ DBC/1012 より、ほとんどのケースにおいて数倍高速であることが分かった。又 GAMMA と比較した場合、優劣が交互して、性能はほぼ拮抗するという結果になった。

2.8.4 では、HDM の商用版（上記の評価は試作機）の性能評価を行った。商用版では種々の改良により試作機に比べて、1.6～2.4 倍性能が向上していることが分った。これと同規模のリレーショナルデータベースシステムで構成した dBase III や SQL Base と性能比較を行った。この結果 HDM はこれらのシステムに比べ 40 倍以上の高性能を達成していることが判明した。

以上行った 3 種類の性能評価結果から HDM は他のリレーショナルデータベースシステムに比べてきわめて高速であることが明らかになった。

２．９．　適用事例

２．９．１　技術情報検索の適用事例

筆者の属する三菱電機（株）情報技術総合研究所では６年ほど前、名菱電子（株）製の商用機“超高速 RDB サーバーHDM-380D”を導入し、情報システム部門によりこの HDM を使った“技術情報検索サービス”が運用されている。このデータベース検索サービスには次のようなものがある。

・社内技術情報検索

・雑誌目次検索

・技術用語検索

・コンピュータ、計測器等の管理情報検索

・特許情報検索（社内／社外等数種）

すべての研究所所員のパソコンないしワークステーションからこれらの技術情報データベースに簡単にアクセスできるため、多くの所員により研究遂行の補助等の目的でこれらのデータベースは日々参照されている。

［１］中間一致検索を基本とし使いやすさが向上

上記のシステムの特徴は“あいまい検索（中間一致検索）”をベースにしていることである。即ちインデックスを使わないで全件サーチを行うことを基本としたデータベース設定としている。これは全件サーチを高速に処理する HDM によって初めて可能になるものである。キーワードの完全一致検索は通常のシステムでもインデックスを使うことにより高速に実行される。又前方一致検索とか後方一致検索といったレベルの“あいまい検索”では、インデックスが利用できるため、通常のリレーショナルデータベースシステムでも、前もって前方一致検索用のインデックスと後方一致検索用のインデックスを作成しておくことにより高速な検索が可能である。これに対し中間一致検索はインデックスが利用できないため、全件サーチが必要になり、通常のリレーショナルデータベースでは非常に時間のかかる処理となる。

注）完全一致検索：キーワードが“abcde”の時、“abcde”と指定して検索する。

前方一致検索：キーワードが“abcde”の時、“abc”等と指定して検索する。

後方一致検索：キーワードが“abcde”の時、“cde”等と指定して検索する。

中間一致検索：キーワードが“abcde”の時、“bcd”等と指定して検索する。

このため通常のリレーショナルデータベースシステムではユーザーにどの種類の検索であるか指定させ、それによって処理方法を変えるのが普通となっている。即ち“中間一致検索”が指定された時にだけ、インデックスが使えないため全件サーチを行うようにして、この場合には非常に時間がかかるようになっている。これに対し、上記のシステムでは中間一致検索を基本として毎回無条件全件サーチを行うため、ユーザーはこれらの検索の種類の指定を行わなくてよい。即ちこのシステムにおいてはユーザーは完全一致検索とか、前方一致検索とか、後方一致検索とか、中間一致検索とかの違いを意識する必要がないために非常に使い勝手がよい。このことを表 2.12 にまとめて示す。

表 2.12　ユーザー指定法の簡単化

種類		検索の内容	通常のデータベースシステム		技術情報検索システム	
			ユーザー指定法	処理速度	ユーザー指定法	処理速度
完全一致検索		ABC というキーワードで検索したい	ABC	速い	ABC	速い
あいまい検索	前方一致検索	最初が ABC で始まるキーワード(ABC･･･)で検索したい	ABC%	速い		
	後方一致検索	最後が ABC であるようなキーワード(･･･ABC)で検索したい	%ABC	速い		
	中間一致検索	“ABC”の文字列を含むようなキーワード(･･･ABC･･･)で検索したい	%ABC%	**非常に遅い**		

↑

簡単

(検索の種類を気にしなくてよい)

［2］中間一致検索ベースでデータベース管理を容易化

データベースを構築する場合には、データベースの入力法に一定の基準を設けるのが普通である。例えば“シソーラスに載っているキーワード以外は使うな”というのも１つの入力基準であるし、“アルファベットは半角で入力すること”等もそうである。このような入力基準を設けることにより、いろいろ多くの人が入力しても、一定の整然としたデータベースとすることが出来、検索の容易化に結びつく。上記表題における“管理”とは主にこのことを言っている。

上記のようにデータベース入力基準の設定は重要な事項であるが、これをどのレベルまで行うかについては、非常に難しい問題が存在する。例えば上記の雑誌目次検索を例にとって,“WindowsNT”というキーワードについて見みてみる。この言葉は、雑誌によって、号によって、ないしは記事によって、“Windows NT”と書いてあったり、”WindowsNT”と書いてあったりする。この場合、この両方を知っている入力者は、どちらで入れていいか迷ってしまう。もし、“これはWindows NTと入力しなさい”とか“WindowsNTと入力しなさい”と言った入力基準があれば上記の迷いは解消される。しかしながら、このレベルまで踏み込んだ入力基準を設定することは、非常に多くの労力を要し、かつその維持、管理も大変である。

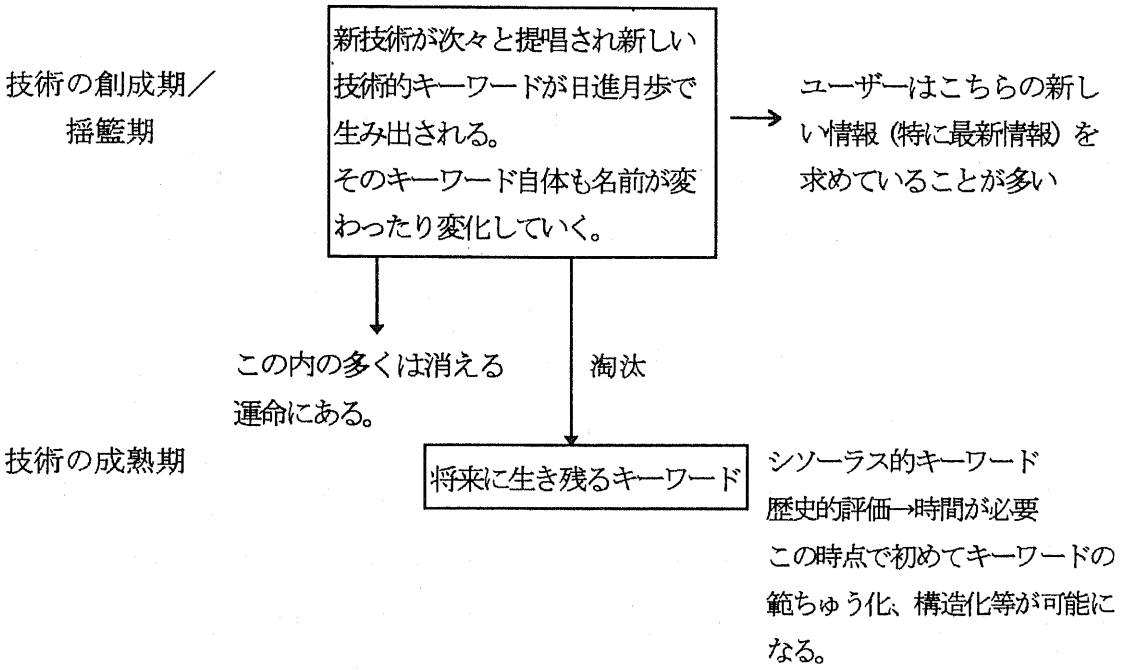


図 2.18　技術的キーワードの経時的変遷

この問題について考察を加えたのが図 2.18 である。新技術が次々と提唱され新しい技術的キーワードが日進月歩で生み出される。そのキーワード自体も名前が変わったり変化していく。例えば、最近デファクトスタンダードになってきた IEEE1394 というインタフェース規格があるが、これは当初 P1394 と呼ばれていた。しかし最近では P1394 という呼び名は全く影をひそめ IEEE1394 と呼ばれるようになった。よりロングレンジの変遷の例では、設計自動化ソフトの範疇を昔は DA（Design Automation）と呼んでいた。これが CAD（Computer Aided Design）と呼ばれるようになって 10 数年続いた後、最近では EDA（Electronic Design Automation）と呼ばれるようになった。

このように出来ては消え、又めまぐるしく変遷する技術的キーワードを管理する方法はあるだろうか。なかなか難しい問題である、というのが筆者の感想である。

この問題に一定の解決法を与えているのが、中間一致検索をベースにした上記の“技術情報検索システム”であるといえる。例えばこの中の雑誌目次検索を例にとれば、そのデータベース入力基準は、

①数字とアルファベットは半角で、その他は全角で入力する。

②入力するデータは出来るだけ雑誌の目次に書かれているそのままの形で入力する。という単純で分かり易いものである。そしてキーワードの管理はいっさい行っていない。データを入力するときは簡単なルールの下にどんどん入れていき、これを取り出すときに若干工夫するようなシステムになっている。（多くの場合“工夫”は必要ないが）。そしてこの工夫がやりやすいシステムになっているというのが特徴である。その第１はキーワードの制約がないため、何等かの要領書を参照する等の必要が無く、検索者は生のデータベースと対峙して自分の思うことをストレートにいろいろやってみて、探すことが出来ることである。第２は中間一致検索を自在に使うことにより、目的のものを見つけやすいことである。例えば前記の WindowsNT の例で言えば“Windows NT or WindowsNT”というように検索条件を設定することにより、両方の形で記述された WindowsNT に関する記事が検索できる。このように素朴に考えて素朴に条件設定して検索できるというのが特徴である。

以上のように HDM の中間一致検索の高速性を利用した設定にすると、キーワード管理の労力の削減が図られる。又インデックスを付けないことから、インデックス設計の手間が省けるという利点がある。さらに、維持管理ではインデックスファイルのダンプロードが不要であるとか、インデックス関連の種々の事項（ファイルのバージョンが正しいか等）からも解放され、データベース作成／管理の労力が大幅に削減される。

［３］検索例

前記の検索の容易さや、このシステムの有効性についての理解を得るために以下に検索例を示す。技術情報検索サービスの中では“社内技術情報検索”が 10 万件規模のリレーションであるため好例となるが、社外秘情報であるため示せない。（この場合 10 秒程度で検索結果が返ってくる。（すべて中間一致検索））このため件数は 1 万件程度と小さいが雑誌目次検索の検索例について説明する。（この場合にはほぼ瞬時に検索結果が返ってくる。）

（１）検索条件の設定

IEEE1394 に関する記事をリストアップしたいが、この単語の“394”という部分しか思い出せなかったケースを想定する。検索結果は発行年月日順にソートして出力するものと

する。この場合には検索条件設定画面に次のように設定すればよい。（下記の検索条件設定画面中“394”と“2”が入力した部分）

```
【 雑誌情報検索 検索条件指定 】
  No.   項   目   名          項   目   別   検   索   条   件   指   定
   1〔目次          〕『394                                          』
   2〔発行          〕『                                             』
   3〔頁            〕『                                             』
   4〔頁数          〕『                                             』
   5〔雑誌名        〕『                                             』
   6〔特集          〕『                                             』
   7〔発行者        〕『                                             』

 条件入力項目番号  『  』
 条件式  『                                                          』
 ソート順『2                                                         』

       条件入力項目番号を設定して下さい。

    [実行:^G]                          [取消:^E]  [終了:^D]  [HELP:^L]
```

（2）検索結果

上記を実行（^Gを入力）すると、以下のように全部で18件の検索結果が、発行日順にソートされた形式で表示される。これらはすべて1394に関するものであり、それ以外のゴミの検索結果は含まれていない。

【 雑誌情報検索 検索結果 】

目次	発行日	頁	頁数	雑誌名
ポストSCSIの設計思想を探る-4部<P1394> マルチメディア用にisochronous機能を備える	940704	152	12	日経エレクトロニクス
ポストSCSIの設計思想を探る-1部 Fiber Channel が先行、SSA と P1394 が追う	940704	126	4	日経エレクトロニクス
3部動画の転送はIEEE1394で決まり、電話や電子スチルカメラをUSB はねらう	950703	95	11	日経エレクトロニクス
パソコン周辺インタフェース、ポストIDE 、ポストSCSIを探る(IDE,Ultra-SCSI,IEEE1394,USB)	951023	141	22	日経エレクトロニクス
IEEE1394をMPEG対応に拡張、標準化団体IECで審議中の仕様の詳細を解説	960129	113	8	日経エレクトロニクス

米 Microsoft 、AV 機器統合する家庭用パソコンを提案、IEEE1394 と USB を標準装備	960422	14	2	日経エレクトロニクス
米 Microsoft 、97 年以降のパソコンの要求仕様を提案、ISA バスは推奨せず、USB や IEEE 1394 を推す(PC97)	960422	7	1	日経エレクトロニクス
ゴッドファーザ現る、IEEE1394 の開発(1)	961021	151	4	日経エレクトロニクス
マルチメディアデータ転送で特許取得、IEEE 1394 の開発(3)	961118	167	4	日経エレクトロニクス
ソニー登場、民生機器寄りの道に、IEEE1394 の開発(4)	961202	165	5	日経エレクトロニクス
家電のセンスを注入し再出発、IEEE1394 の開発ストーリ（第 5 回）	961216	167	3	日経エレクトロニクス
HDD メーカに見放され窮地に、IEEE1394 の開発（6）	970106	143	4	日経エレクトロニクス
次世代インタフェースが登場、USB と IEEE1394、普及は今夏以降	970106	64	4	日経コンピュータ
米 Apple から、業界団体に主役交代、IEEE1394 の開発(7)	970113	129	4	日経エレクトロニクス
普及期迎え、開発者はそれぞれの道へ、IEEE 1394 の開発（最終回）	970210	169	5	日経エレクトロニクス
第 3 部－USB(Universal Serial Bus) はどうして 1394 より安くできるのか	970601	158	14	日経バイト
＜特集＞パソコンにデバイスベイが現れる、パソコンの拡張スロットが変わる、PCI バススロットからデバイスベイへ、IEEE1394 と USB を採用	970630	89	1	日経エレクトロニクス
＜特集＞離陸体制整う IEEE1394、ウィンテルのサポート本格化	970908	99	1	日経エレクトロニクス

---------- (18/18)

この検索結果は大きく以下の２つの目的で利用できる。

①記事抽出のための目次として

例えば IEEE1394 を勉強したい場合には、例えば検索結果の１番目の記事が良さそうと

分かるので、日経エレクトロニクスの1994年7月4日号を捜しp152を開けばよい。この他に例えば、MPEGとの関わりを知りたいときには、同様にして5番目の記事を持ってくればよいことが分かる。

②流れを知る

この検索結果全体をながめるとIEEE1394は1994年頃から言われ始めた言葉で1997年現在も記事になっている技術キーワードであるという大きな流れがつかめる。（注：このデータベースには1989年からの記事が入っている。）特に1996年には6件と多くの記事になっており、1394がデファクトスタンダードに向けて大きく発展した年であることがうかがえる。又この目次のリストを年代順に見ていくことにより、その移り変わりの様子、例えば当初ポストSCSIという位置づけだったものが、徐々に家庭用マルチメディアという方向に動いてきている傾向などが読みとれる。

以上のように、この検索結果をながめるだけでもIEEE1394の概要がかなりよく分かってしまうことが、このシステムの特徴となっている。

（3）検索条件の設定

今度はIEEE1394に対して“39”という部分しか思い出せなかった場合の検索例である。（これは正真正銘の中間一致検索である。）この場合にも、以下のように検索条件の“目次”欄に“39”を、“ソート順”欄に“2”（発行年月日順）を入力する。

【 雑誌情報検索 検索条件指定 】

No.	項目名	項目別検索条件指定
1	〔目次 〕	『39 』
2	〔発行 〕	『 』
3	〔頁 〕	『 』
4	〔頁数 〕	『 』
5	〔雑誌名 〕	『 』
6	〔特集 〕	『 』
7	〔発行者 〕	『 』

条件入力項目番号 『 』
条件式 『 』
ソート順『2 』
条件入力項目番号を設定して下さい。

[実行:^G] [取消:^E] [終了:^D] [HELP:^L]

（4）検索結果

【 雑誌情報検索 検索結果 】

目次	発行日	頁	頁数	雑誌名
パソコン・マイクロチャネル対応の 32 ビット・パソコンを 39 万 8000 円で発売	891211	119	1	日経エレクトロニクス
厚さ 39mm の家庭用ファクシミリ	901029	183	10	日経エレクトロニクス
Little Language---- 自由きままに Shell プログラミング(39)-Korn シェル	920101	58	5	UNIX MAGAZINE
システム/390 パソコンが本格出荷、2000 年問題の安価な検証マシンに	960401	46	3	日経コンピュータ

（上記の 4 件以外は前例と同じ内容のため省略）

(22/22)

この場合には検索結果は 22 件となり、4 件のゴミ情報が混じってくる。これは 1394 とは関係ない、“39”という文字列を含む別の記事も検索されてしまうためであり、仕方がない。（例えば“厚さ 39mm の～”といった記事も一緒に検索されてしまう。）このように 18 件に対して 4 件程度のゴミであれば検索者がふるい落とすことは容易である。以上、IEEE1394 というキーワードに対して“39”という条件を設定するようなきわめて“あいまいな”検索に対しても、この検索システムは高速に十分な結果を出してくれることが分かる。

２．１０　結言

多様化、大容量化するデータの高速検索に対するユーザーニーズ、とりわけ非定型データの高速全件検索は、現在の汎用的なソフトウェアデータベースでは不得手な分野である。このニーズを満たすことを目的として、リレーショナルデータベース処理を専用に高速実行するデータベースマシンのアーキテクチャを提案し、試作・評価した。

本章では最初にこの試作したデータベースマシン HDM（High-speed Database Machine）の基本的な設計思想について述べ、次いで CPU とディスクを複数用いて効率的に並列処理を実行するハードウェアの方式的特徴について記述した。されに、このハードウェア上でリレーショナルデータベース処理を高速に実行するために種々工夫を加えたアルゴリズム、そのアルゴリズムを実現するために開発した各種ソフトウェアの構成と働きについて述べた。次に、このマシンの高速化のためのもう１つの重要な要素となっているディスクアクセス方式の特徴について記述した。続いてこのデータベースマシンで実現している高信頼化機能として、障害回復機能について記述した。次に試作機及び商用版の性能評価結果について記述し、最後に適用事例について記述した。

性能評価結果から、このデータベースマシンは他の種々のデータベースシステムに比べてきわめて高性能であることが明らかになった。例えば、商用版の性能評価では他の同規模のデータベースシステムよりも約 40 倍高速であることが示された。又適用事例では、HDM の高速性故に、ユーザー指定を細かく行わなくても良いように出来るため、ユーザーインタフェースが簡単化出来るという利点を示した。又インデックスが不要、又は少なくてよいため、データベースの設計／管理の手間も大いに省ける利点があることを示した。

第３章　基本型・分散ＲＡＩＤ方式ビデオサーバーの提案と試作

３．１ 緒言

近年のマルチメディブームによりＶＯＤ（Video on Demand）の研究が盛んになってきている。しかしながらWWWの使用経験等から、電話回線等のワイドエリアネットワーク・インフラの能力不足が見えてきており、広域を対象としたＶＯＤの実現には今しばらくの時間が必要な状況となっている。

これに対しＬＡＮ等を利用した局所的なネットワークにおいては、100Mbit-Ethernet、Giga-Bit-Ethernet等の普及が進むことにより、近い将来ＶＯＤが広く普及する可能性がある。この場合の利用形態であるが、よく言われるようなホテル内 VOD システムとか、旅客機内 VOD システムといった特定の応用に特化したシステムの他にも、一般のオフィスＬＡＮにおける一つの応用（application）として使われる可能性も十分にある。後者のケースを想定した場合に、ビデオサーバーに要求される条件としては、価格的に十分安価であることが１つと、通常のオフィスＬＡＮ環境に大きな変更を加えることなくスムーズに適用できることの２点が重要となってくる。

本研究はこの方向を目指したビデオサーバーの最適な方式（価格性能比が）を求めることを目的としている。本章では最初にEthernet上でビデオ配信を行う場合の基礎データを得るための各種実験を行った結果について述べる。次にこの実験結果を十分に考慮し、そこで示された最大のボトルネックを解決する手段として、分散ＲＡＩＤ方式ビデオサーバーの１方式を提案する。この方式のビデオサーバーの試作評価を行った結果、きわめて高性能かつ価格性能比の良いビデオサーバーが実現出来ることが実証された。この方式によれば、普及型の安価なサーバーを増設していくことにより、その台数に比例した性能向上が図られるため、きわめて良好な価格性能比が得られる。又ここで試作した方式は、ハー

ドウェアは普及型の汎用部品だけで構成されているため安価であり、一方ソフトウェアの面でも通常の Windows 環境に対して、若干のソフトウェアの追加を行うだけでよい（導入容易性）という２つの大きな特徴がある。これは上記で述べたオフィスＬＡＮ向けビデオサーバーの適性（安価、導入容易性）にかなうものである。

３．１．１ 前提

映像や音声を扱う場合には当然のことながら実時間性が要求されるため、IEEE1394 のように Isochronous 転送と呼ばれる、マルチメディア用にリアルタイム伝送を可能としたインタフェースが提案されている。ビデオサーバーにおいてもこの点（リアルタイム伝送）を重視して研究されているものがある。［Tobagi-93］これらのビデオサーバーにおいては、クライアントで放映中の映像／音声データの枯渇(starvation)が起きないように、サーバーから同期的にコンスタントにデータを送るように工夫がなされている。

これに対し特にリアルタイム伝送ということは意識せず、従来のファイルデータの転送と同様な形でデータ転送する方式のビデオサーバーも提案されている。［堂之下-95］［岡-95］　前者の場合でも例えばＬＡＮを他の系（例えば一般使用の FTP(File Transfer Protocol)や電子メール等）と共用していて、他の系が多量にデータ転送を始めたような場合にはビデオ系の転送容量が減少してデータ供給の同期性は保たれなくなる訳であり、伝送路の負荷がクリティカルになってきた場合には、いずれにせよリアルタイム伝送が厳しい状況となる。そのため伝送路にはある程度余裕をもたせた設計値を設定することとなるが、そのような状況においては後者の方式でも十分使いものになるのではないかという考え方が成り立つ。本研究では後者の方式を念頭に置き、ごく通常使われている環境、即ちＮＦＳサーバーやＬＡＮマネージャーにファイルアクセスする形式で、クライアントがビデオデータを得る方式を考える。後述のように、この方式で十分良質なビデオストリームが供給出来ることが確認出来た。

３．２　ＬＡＮ上のＣ／Ｓ環境におけるビデオサーバー評価実験

まず最初にビデオデータの伝送に関わる各種の特性を調べるため、Ethernet-LAN 上で通常のサーバーをビデオサーバーとして設定した環境上において評価実験を実施した。

３．２．１　基本モデルの解析

図 3.1 は LAN 接続型ビデオサーバーの基本モデルを示す。この図は LAN に接続された複数のクライアントが、同一 LAN 上の１台のビデオサーバーに同時にアクセスし、それぞれが所望のビデオを鑑賞する流れを示している。各クライアントはビデオサーバーに対し、所望のビデオプログラムのファイルの読み出し要求を出し、このデータを受け取るとデータの伸長を行った後、モニターに動画を表示したり音声を再生していく。

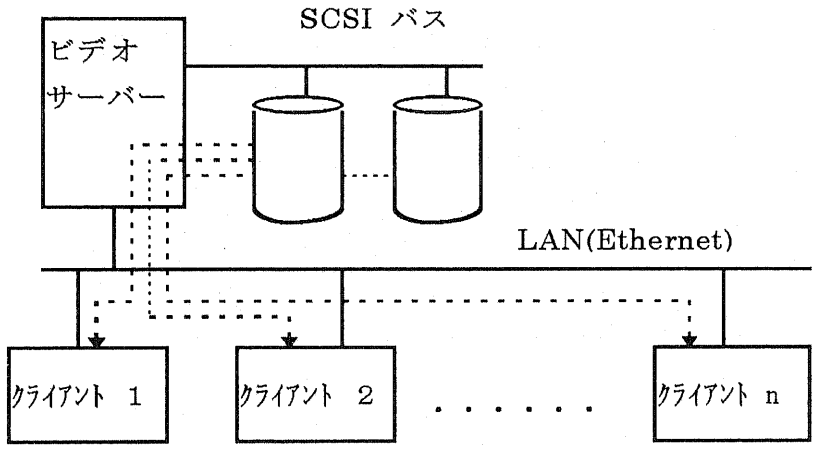


図 3.1　ビデオサーバーの基本モデル

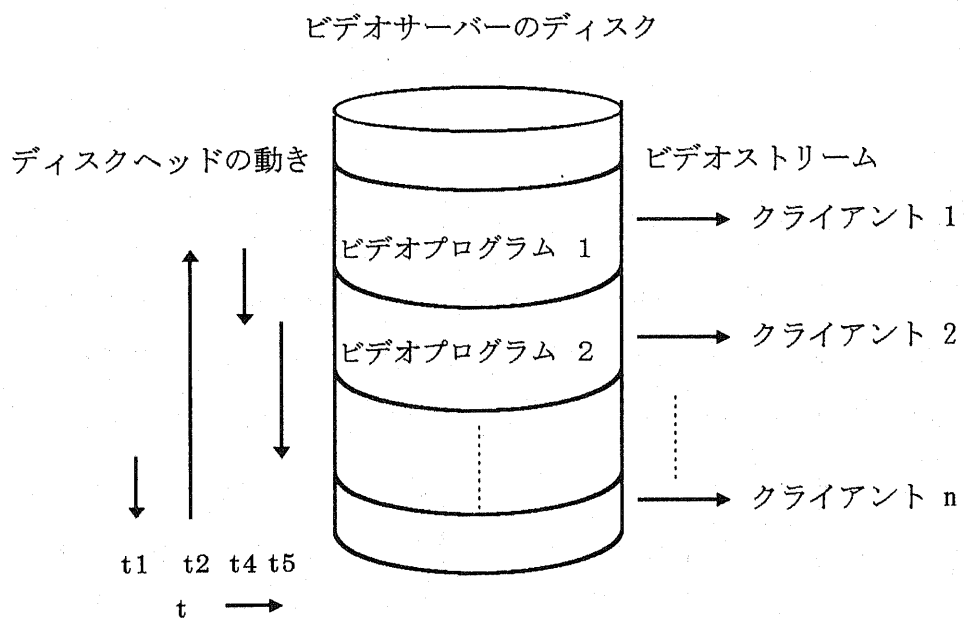


図 3.2　ビデオサーバーのディスクの動き

図 3.2 はこの時のビデオサーバー中のディスクの動きを示す。1 つのクライアントだけがビデオ鑑賞しているのであれば、サーバーのディスクヘッドの動きはシーケンシャルでよいが、ビデオサーバーにおいては複数のクライアントが同時アクセスしているのが普通であるから、図 3.2 に示されるようにサーバーのディスクのヘッドは目まぐるしく動くことになる。1 つのビデオサーバーから少しでも多くの本数のビデオを同時供給させようとした場合、例えばまずこのようにディスク性能が問題となりそうなことが分かるが、それ以外にもどのような問題があるか以下に全体的な解析を試みる。

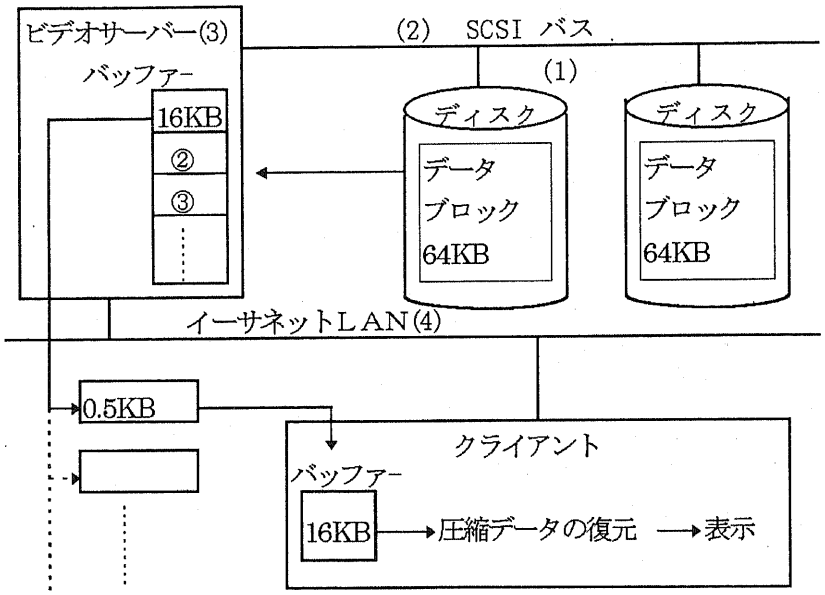


図 3.3 データの流れと性能ボトルネック

ビデオサーバーの性能とは、如何に多くのビデオストリームを複数のクライアントに同時にリアルタイム（良好な画像／音声で）に供給できるかと言う点に集約出来る。（ビデオストリームとは 1 つのクライアントに対して送られるビデオプログラムのデータの流れを指す）。ビデオサーバーシステム全体としてみた場合にはこのビデオサーバーの供給能力の他に、通信路の伝送能力も問題となってくる。図 3.3 はこの通信路も含めたビデオサーバーシステム全体の性能を解析する図である。図 3.3 においてディスク上のビデ

オデータはブロック単位（例えば64KB）に読み出され、サーバーの主メモリ上のクライアント毎に用意されたバッファにストアされる。クライアントから一時に来る要求は通常このブロックより小さなものであるため、上記バッファーからその部分が切り出されてクライアントに転送される。Ethernetの LAN上ではこれがさらに0.5KBないし1KBの単位に分割されて転送が行われる。Ethernetからデータを受け取るとクライアントはこれをデータ伸長し画面に表示するが、この部分についてはクライアントの範疇であるため本研究では扱わない。クライアントは単に 1 本のビデオを自分のモニタに放映するだけであり、例えば自身のCD-ROMから読み出して放映するのと変わらない。このため性能上の問題は特に存在しない。本研究が対象とするのは、多数のクライアントに多数のビデオストリームを同時供給する、サーバー及び通信路の性能問題についてである。

さて上記の図3.3を用いた動作解析から、ビデオサーバーの性能（ビデオストリーム同時配信数）を決定する要因としては以下の4点に集約出来る。

(1)サーバーのディスクの性能

(2)このディスクがつながるSCSIバスの性能

(3)サーバーのCPU性能（メモリ性能等も含む）

(4)LANの性能

これらの4点に着目して、以下にLAN接続型ビデオサーバーの性能決定要因について調べる実験を行った。

３．２．２　実験システムの構成

図3.4にこの実験で用いたビデオサーバーシステムの構成図を示す。この構成はごく普通のクライアントサーバー環境に他ならない。図3.4においてサーバーは通常のファイルサーバーの役割をしているに過ぎず、クライアントはNFS(Network File System)を使ってサーバーからビデオデータを取り出し、これを自らのモニターに放映する。これだけ見れば単にファイルサーバーの性能評価を行うということになるが、転送するデータがビデオデータでありそれが実時間で供給されるか否か、即ち途切れのない良好な映像／音声が得られるかが問題になるところに違いがある。なおサーバーとしては3.2.4節まで

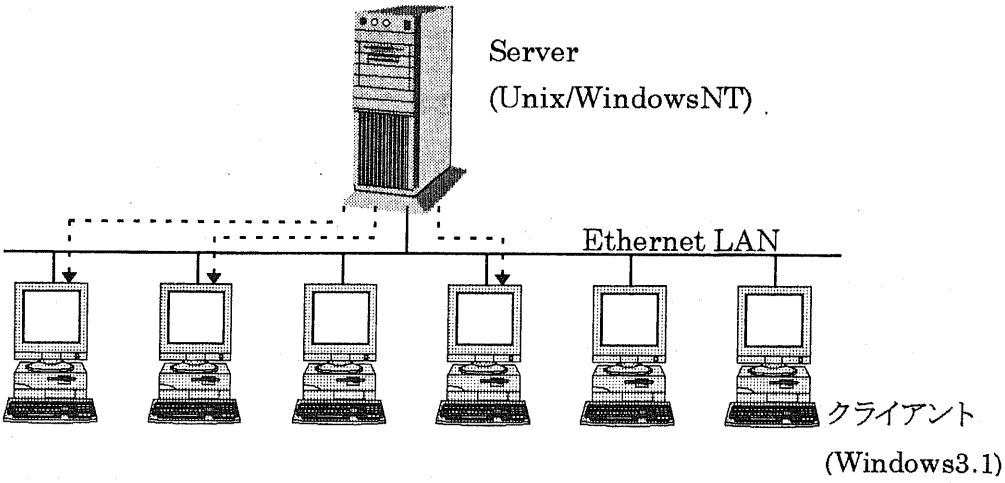


図 3.4　実験システムの構成

はUNIXサーバーを、3.2.5節からはWindowsNTサーバーを使用した。ビデオプログラムは約5分の長さのものを使用し、サーバー上ではこれを実クライアント数だけコピーして、それぞれ実クライアント毎に物理的に異なる所にアクセスするようにした。実際に使用したクライアントの数は6台であるが、それ以上のクライアント数の試験を行うために、1台の実クライアントがあたかもクライアント数台分のふるまいをするような疑似クライアントプログラムを作成・使用した。以下にこの実験に使用した機器、ソフトウェアを示す。

＜サーバー＝UNIX＞
　　ME/R7150-50(PA-RISC,50MHz)
　　主メモリ：64MB
　　OS：HP-UX9.0,NFS(UDP/IP)
＜サーバー＝WindowsNT＞
　　FT//ex(Pentium,90MHz)
　　主メモリ：32MB
　　OS：WindowsNT3.5,LAN Manager(NetBEUI)
＜サーバーのディスク（代表例）＞
　　IBM SPITFIRE　5400rpm
　　Single cyl seek time=0.6ms(R)-2.5ms(W)
　　Full stroke seek time=16.5ms(r),18ms(W)
　　Data transfer rate from media=6-5MB/s

sector size=512B, sector/track=108-90
# of data heads=5,# of U-cylinder=4119
formatted capacity:1.05GB

＜クライアント＞

apricot M3466-A154W(Pentium,90MHz)
主メモリ：16MB
OS：Windows3.1
NFS:ChameleonNFS
LAN Manager
Video for Windows

＜LANカード＞

Ethernet-EISA：3Com 3C579
Ethernet-PCI　：B8832
100VG-AnyLAN　：HP J2585A PCI Adapter

＜ビデオデータ＞

方式:　AVI、フレーム数＝5128フレーム、342秒
ビデオ:　cvid 320x240x24、 15.000fps、 50881976バイト
オーディオ：モノラル、11.025Khz、8ビット

＜フレーム落ち測定プログラム＞

vidtest.exe（Vieo for Windowsに付属）

＜疑似クライアントプログラム＞

今回の実験のために作成したクライアント上で動くプログラム。Video for Windowsと同様にサーバーに対しビデオデータの要求を行い、データを読み出す。1台のクライアントから、あたかも多数のクライアントがあるかのようにサーバーに対してデータ要求を出すため、サーバー側から見るとクライアントが増えたように見える。

３．２．３　ディスク・ボトルネック

最初に第1の点、即ちディスク・ボトルネックについて調べる実験を行った。まず普通によくある小規模サーバー環境、即ちサーバーのディスク台数＝1、データのブロックサイズ＝8KBという設定でビデオ放映実験を行い、クライアント数を1～5に変化させた時の映像のコマ落ち率を測定した。ここでコマ落ち率とは、再生されるべき全フレーム中、データの遅れにより再生されなかったフレーム数の割合であり、上記フレーム落ち測定プログラムにより“frame drop rate”として表示されるものである。図3.5にこの測定結果を示す。

筆者の実体験によれば、コマ落ち率が1%を越えると人間の目、耳にも質の低下がはっ

きり分かるようになる。図 3.5 の測定結果では 4 クライアント以上になるとコマ落ち率がこの 1%を越えている。このことからこの場合 n は 3 クライアント（=3 ストリーム）までが限界であると言える。このことは翻って考えてみれば、ごく普通に存在する小規模な LAN 上のクライアント・サーバ環境においてでも、ファイルサーバーにビデオデータを何本か入れておけば、3 人まではそれぞれ任意のビデオを同時鑑賞出来るということである。（即ち小規模なビデオサーバーとなることが出来る。）

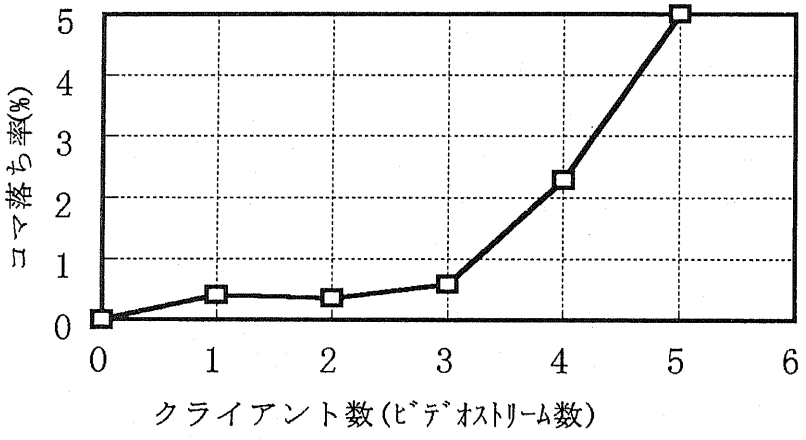


図 3.5 ディスク・ボトルネック

さてここで用いた 5 分間の長さのビデオプログラムの他に、別途 45 秒（7 MB）の短いビデオプログラム（1 本）を使った実験も別途行ったが、この時には 6 クライアントまで良質な再生が出来た。これは 7MB の映像データが常にディスクキャッシュに入ってしまいビデオデータが毎回主メモリ上のディスクキャッシュから読み出されるためと考えられる。このことから図 3.5 で 3 ストリームまでが限界となったのはディスク・ボトルネックが原因と推定できる。

表 3.1　ディスク台数とブロックサイズ

ディスク台数	ブロックサイズ（ＫＢ）	クライアント数	コマ落ち率（%）
1	8	5	5.0
2	8	5	0.374
1	64	5	1.17
2	64	5	0.335

このディスク・ボトルネックの問題を解決するために以下のような対策を実施して再測定を行った。この測定結果を表 3.1 に示す。

ブロックサイズ：8KB→64KB

ディスク台数　：1 台→2 台

表 3.1 の測定結果から、ディスク台数を 1 台から 2 台（ソフトウェアストライピング）に増設するとコマ落ち率は 5%から 0.374%に改善されてディスクネックが解消されることが分かる。これはクライアントからのディスクアクセス要求がディスク 2 台に分散されて、ディスク 1 台当たりの負荷が 1／2 に減少するためと推定できる。一方、ブロックサイズを 8KB から 64KB に増やすことによっても、コマ落ち率は 5%から 1.17%に改善しディスクネックが緩和されることが分かる。これもブロックサイズが 8 倍になった結果ディスクアクセス回数が 1／8 に減少したためと推定出来る。

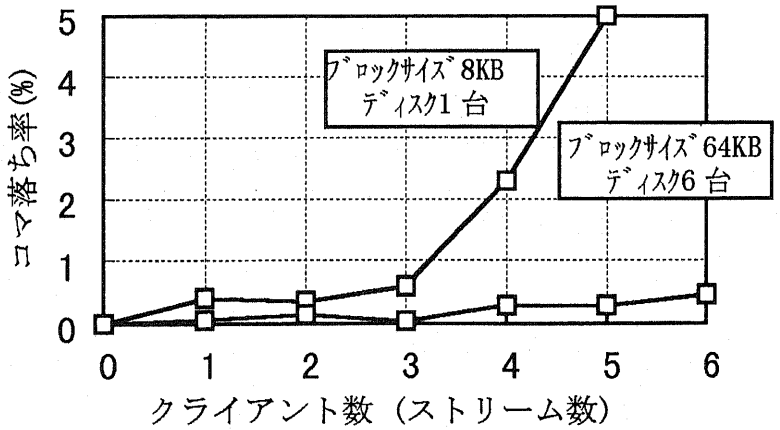


図 3.6 ディスク・ボトルネックの解消

さらに、ディスク台数を 6 台に増やしてソフトウェアストライピング構成とし、ブロックサイズを 64KB に設定して再度測定した結果を図 3.6 に示す。図 3.6 において下側の線がこの測定結果でありであり、上側の線は図 3.5 の内容を比較のために示している。上側の線から下側の線にコマ落ち率が大きく改善されていることが分かる。以上のようにディスクネックの問題は、ディスク上のデータのブロックサイズの拡大や、ディスクの増設により解決出来ることが明らかになった。

３．２．４　ＳＣＳＩバス・ボトルネック

次に第２の点、即ちSCSIバス・ボトルネックを調査する目的で次のような実験を行った。64KBでブロッキングされたディスク上のデータをシーケンシャルに読み出すプログラムを作成し、これをサーバー上で動かし、そのデータ転送レートをディスク台数を変化させて測定したものが図3.7である。ここでディスクが2～6台のケースでは1本のSCSIバスに接続してソフトウェアストライピングを施している。

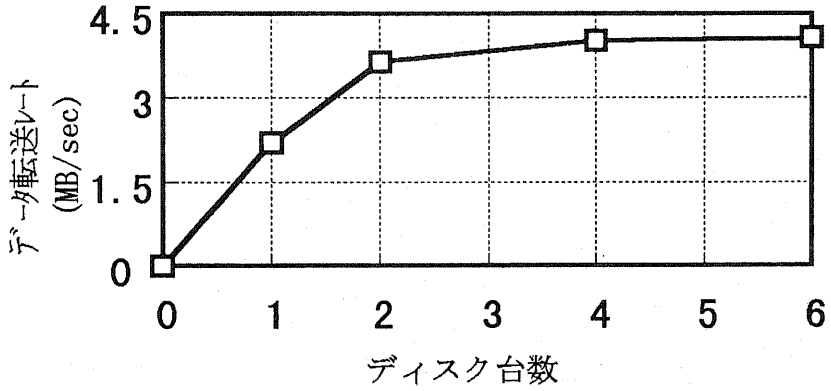


図3.7 SCSI ボトルネック

図3.7から同一のSCSIバスに接続するディスクがn台に増えてディスク側のデータ供給能力がn倍になっても、全体のデータ転送レートはリニアに増加しないことが分かる。ディスク１台で2.25MB／sec、２台で3.75MB／secと段々飽和して行き、約4MB／secのところで飽和する。このようにリニアに伸びていかない理由の１つは同一SCSIバスにつながる複数ディスク間の競合の問題による。あるディスクを起動しようとした時にSCSIバスが他のディスクで使用中のため待たされたり、ディスクがデータ転送を開始しようとする時に、他のディスクがSCSIバスを使用中のため待たされたりという競合が生ずるためである。もう一つの理由はサーバー側にあるSCSIコントローラの処理能力の限界によるものであり、図3.7において10MB／sec規格のSCSIバス上であるにもかかわらず4MB／secで飽和してしまっているのがこれにあたる。SCSIバスの占有時間は起動／終了処理に比してデータ転送処理が大きいことから、図3.7の結果をデータ転送レートの面からビデオストリーム転送に近似して考えてみると、安全サイドに見積もって、ディスクが２台のところ、即ち24ストリーム＝28.8Mbps＝3.6MB／sec.程度が１本のSCSI

バスの許容能力ではないかと推定できる。この値がディスクのそれ（6 ストリーム以上）に比して十分高いこと、又サーバー上で SCSI バスの本数は容易に増設出来ることから、この SCSI バス・ボトルネックは当面の問題とはならないことが結論づけられる。

３．２．５　ＣＰＵボトルネック

次に第 3 の点、即ちサーバの CPU 性能（主メモリ性能を含む）によるボトルネックを調べる実験を行った。なおここからはサーバーは UNIX（NFS)から WindowsNT（LAN Manager）に変更した。

［１］EISA-LAN ボード

サーバーの LAN コントローラに EISA 接続タイプのものを使用して、ビデオ放映実験を行いコマ落ち率を測定した結果が図 3.8 である。図 3.8 では 5 ストリーム放映時のコマ落ち率が 1%以下と表 3.1 の同条件の場合と比べて良くなっているが、これは UNIX と WindowsNT の環境の差違に起因する。一方、図 3.9 は図 3.8 に対応したサーバーの CPU 使用率を測定したものである。

測定条件：ディスク台数　　＝1

　　ブロックサイズ　＝64KB

　　LAN ボード　　＝EISA 接続型

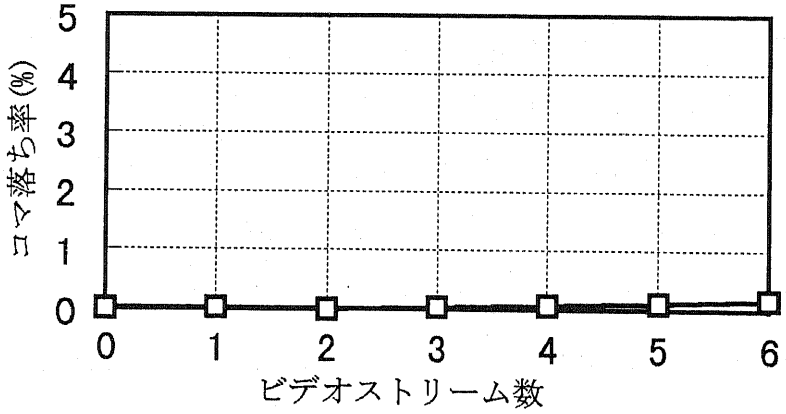


図 3.8 コマ落ち率（EISA)

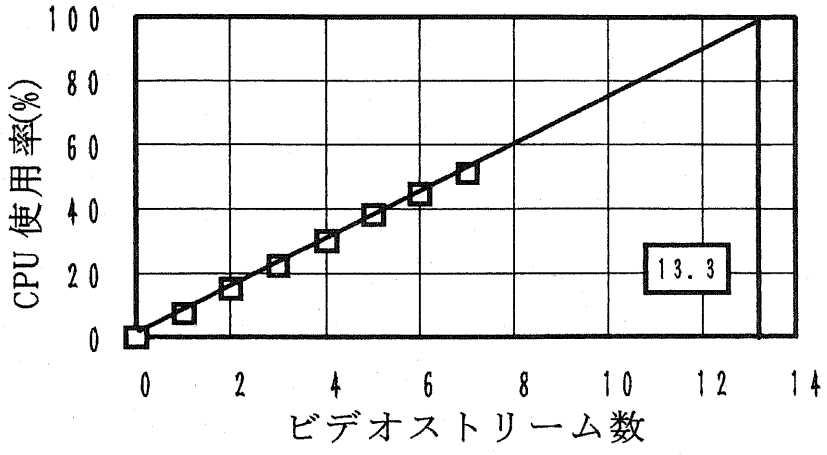


図 3.9 CPU 使用率 (EISA)

図 3.9 において CPU の使用率はきれいにグラフの原点を通る直線上に乗っている。即ちビデオストリーム数に比例して CPU 使用率も増加することが分かる。図 3.9 において CPU 使用率が 100%になるところは 13.3 ストリームであり、これが CPU 使用率の面から見た供給可能ストリーム数の上限と予測できる。

［2］ PCI-LAN ボード

LAN ボードを PCI バス接続タイプのものに変えて上記と同様の測定を行った。この結果を図 3.10、図 3.11 に示す。この場合も図 3.10 は 5 ストリームまでは良好な映像である

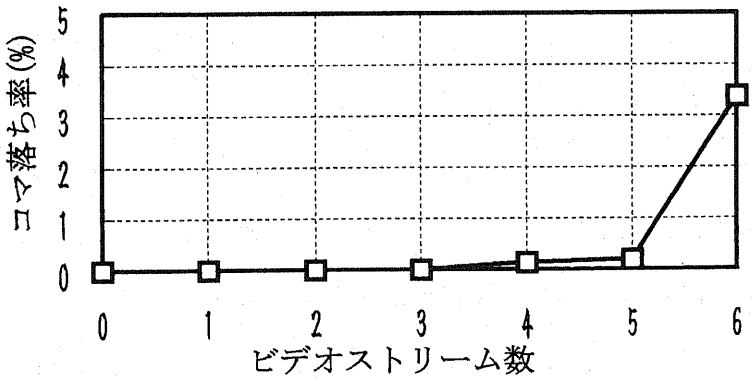


図 3.10 コマ落ち率 (PCI)

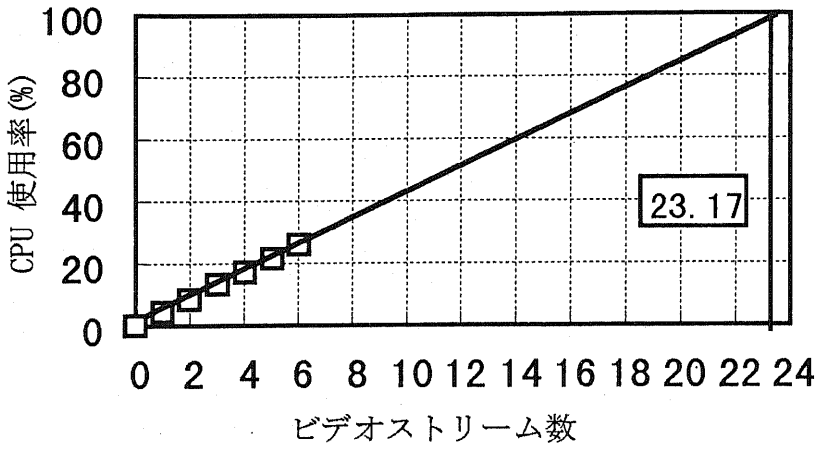


図 3.11 CPU 使用率 (PCI)

ことを示しており、図 3.11 の CPU 使用率のグラフはグラフの原点を通る直線上に乗っている。これから CPU 使用率の面からみた供給可能ストリーム数の上限は、23.2 ストリームと予測出来る。この値は EISA ボードの場合の 13.3 ストリームと比べて 1.74 倍も改善されており意外な結果である。

この違いは EISA-LAN ボード用のドライバプログラムと PCI-LAN ボード用のドライバプログラムの処理内容の違いに起因すると考えられる。即ち PCI－LAN ボードを使った方が LAN 制御のための CPU 負荷が 1／1.74 と少なくて済むことを意味する。サーバーが LAN を経由してクライアントにビデオデータを転送する時、実際に主メモリとの間でデータ転送を行うのは LAN ボードであって、サーバーの CPU はこの起動／終了処理を行うにすぎない。このためにとられる CPU 時間が、LAN ボードを変えただけで 1.74 倍も異なるということは、見方を変えればサーバーの CPU が LAN 制御のために使われる時間の多さを物語るものと言える。

さて前記のディスク・ボトルネック、SCSI バス・ボトルネックの場合は、増設という手段で解決出来た。これに対しサーバーの CPU は増設する訳にはいかない。このための解決策としては単純にサーバーCPU の強化を図るしかないが、この方法としては(1)サーバの CPU をより高性能なものに置き換える、(2)マルチ CPU 型のものに置き換える等がある。これらの手段はいずれもシステム価格を上昇させるが、最近はグロシュの法則の逆が真とも言われているように［高橋-85］、CPU の高性能化はお金を掛けた程には効果

が上がらないことが多い。さらにLAN制御プログラムはメモリコピーが多いのが欠点と言われているが、メモリコピー性能は主メモリ性能が主に効いてくるため、単にCPUの強化だけでは効果が上がらないという問題もある。

以上のようにサーバーのCPU能力については有効な増強手段が見あたらず、ビデオサーバーの性能向上を図る上で大きな障害となることが明らかになった。

［3］3本のLANを接続

上記の図3.9、図3.11においてCPU使用率の面からの供給可能ストリーム数上限の予測を行ったが、この予測の妥当性を検証する目的で以下のような実験を行った。即ちEISA接続タイプのLANボードを3枚用いサーバーにLANを3本つないだ構成とし、

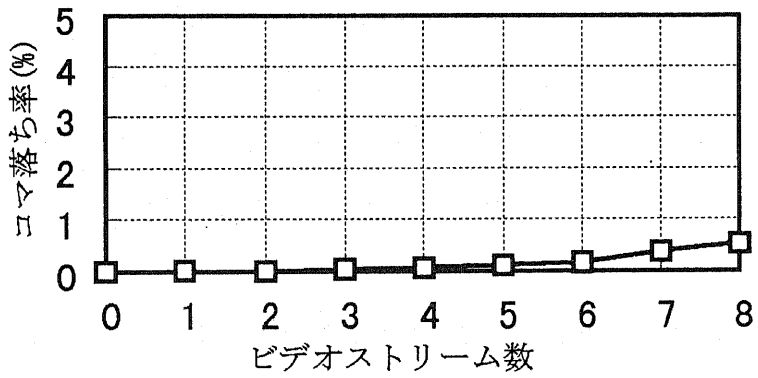


図3.12 コマ落ち率（LAN3本）

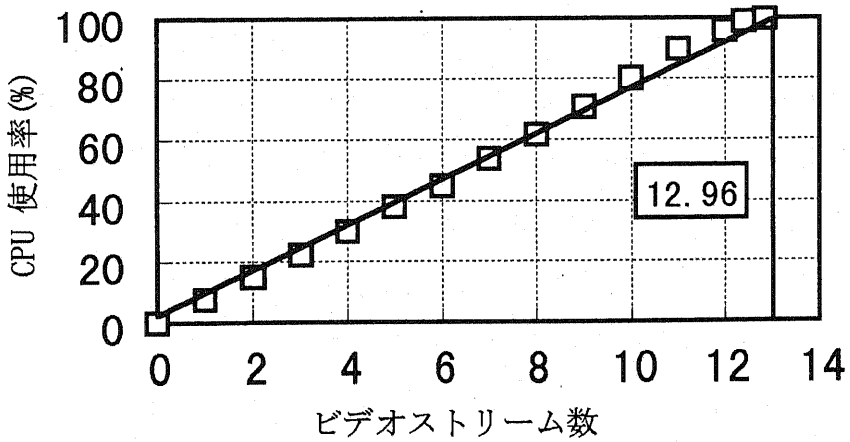


図3.13 CPU使用率（LAN3本）

各LANにはそれぞれクライアント2台、2台、及び1台をつなぎ、1台のクライアントのみビデオ放映を行い、他の 4 台のクライアントでは疑似クライアントプログラムを走らせた。

この測定結果を図 3.12、図 3.13 に示す。LAN 容量が 3 倍になったことで図 3.12 に示されるように非常に良好な映像が得られていることが分かる。なおこのシステム構成では 1 つのクライアントでビデオを放映しながらの測定では 8 ストリームまでが限界であったため（注：この限界を除くにはクライアントがもう 1 台必要）、図 3.13 における 9 ストリーム以上の測定点では 5 台のクライアントすべてにおいて疑似クライアントプログラムを走らせている。この図 3.13 と以前に測定した図 3.9 を比較すると、供給可能ストリーム数上限は図 3.9 の 13.3 ストリームに対し図 3.13 では 12.96 ストリームとほぼ同様の値である。又グラフの形も両者共同じ傾きの直線となっている。このことから図 3.9 の時に行った予測、即ちストリーム数と CPU 使用率が比例すること（CPU 使用率が 100%に近い領域でも）、及び供給可能ストリーム数上限値の予測が正しかったことが検証されたことになる。

又 LAN 増設を行った場合に LAN の本数に比例して供給可能ストリーム数が増加することが実システム上で検証出来たので、LAN 増設については特に問題がないことが実証出来た。

３．２．６　ＬＡＮボトルネック

最後に第 4 の点、即ち LAN 性能によるボトルネックを調べる実験を行った。これについてはLANを通常のEthernet(10Mbps)から100Mbps-Ethernet(100VG-AnyLAN)に変えて測定し両者を比較する方法をとった。

測定条件：ディスク台数　　＝3（ハードウェア RAID）

ブロックサイズ　＝64KB

LAN　　＝100VG-Any LAN

LAN ボード　　＝PCI 接続型

図 3.14 の下側の線がこの時のフレーム落ち率の測定結果を示している。上側の線は図

3.10 の 10Mbps-Ethernet での測定値である。100Mbps-Ethernet になると 4 ストリーム以上のところで映像品質が飛躍的に改善されることが分かる。即ち単に LAN を変えただけで映像品質がこのように顕著に向上する。このことは映像品質確保のための通信路の重要性を如実に示している。一方図 3.15 はこの時の CPU 使用率を示しているが、

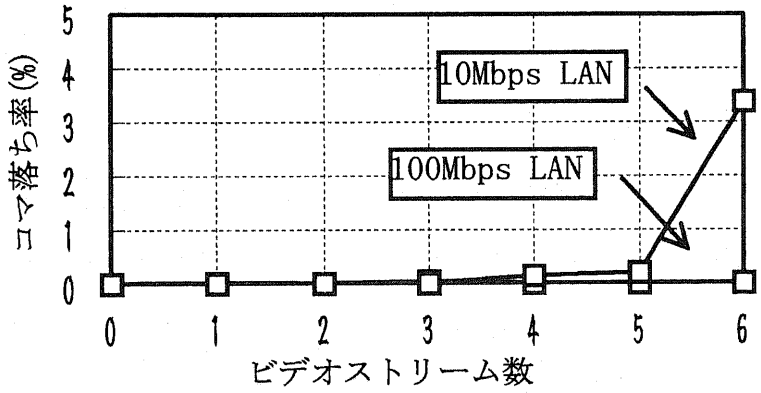


図 3.14 コマ落ち率（100Mbps LAN）

10Mbps-Ethernet の図 3.11 の場合と同じ上限値（約 23 ストリーム）を示している点が興味深い。これは前述のように CPU は起動／終了処理だけを行い、データ転送そのものにはノータッチであるため、起動と終了の間に実行されるデータ転送の速い／遅いの差は、CPU 自身の処理時間の多い／少ないには直接影響して来ないためと考えられる。

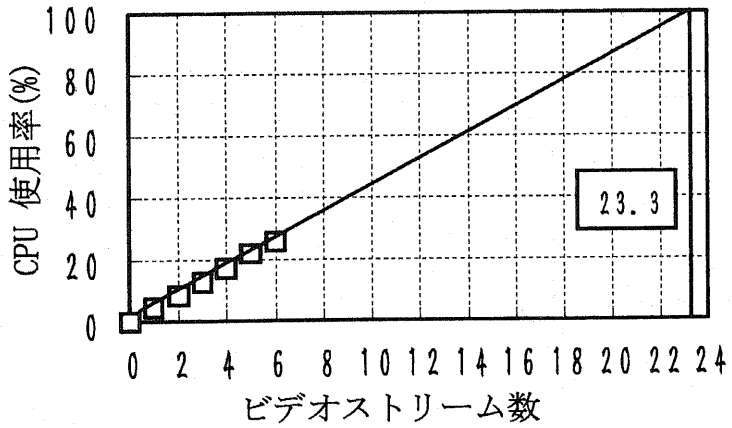


図 3.15 CPU 使用率（100Mbps LAN）

以上のように映像品質確保のためにLAN性能がきわめて重要なことが分かったが、前述の３．２．５[3]の実験で実証されたように、LANボトルネックに対してはLANの増設という解決手段が残されている。

３．２．７　実験結果のまとめ

以上の測定結果からビデオサーバーにおけるボトルネックとして、ディスク、SCSI、LANについては増設という対処手段があるが、サーバーCPUについてはそれがなく、問題になりそうなことが分かった。図3.16は以上に行った実験結果を概略的にまとめて表したものである。各要素毎の概略ビデオストリーム許容数と増設手段の有無を示している。例えば図中のディスクのところの矢印図形１つはディスク１台を意味し、その中に書かれている６ストリームとは、ディスク１台で６ビデオストリーム供給可能なことを表している。矢印の方向はデータの流れる方向を示している。なお図3.16において各要素毎のビデオストリーム許容数は次のように設定した。ディスク１台のビデオストリーム許容数は図3.8から少なくとも６ストリームまで良好であるため６とした。SCSIの許容能力については図3.7から＝24ストリームと予測した。CPUについては図3.11の結果から23ストリームとした。LANについては、図3.14の10Mbps-LANでは５ストリームが限度なので、100Mbps-LANではこれを10倍して50ストリームとした。

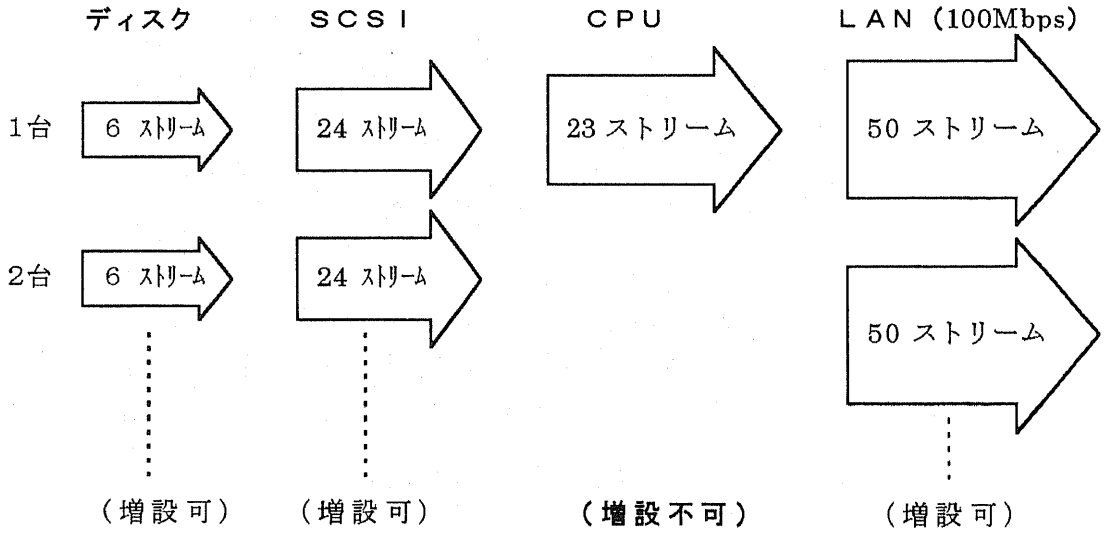


図 3.16　各要素のビデオストリーム許容数と増設可否

図 3.16 を使うことにより、ビデオストリーム数最大許容値（仕様）が与えられた時、それぞれの要素を何個ずつ用意すればよいかが概算できる。例えばビデオストリーム数最大許容値が 12 であればディスクは 2 台用意すればよい。図 3.16 に示されるように、CPU については**“増設不可”**のため 1 個しか使えず、その許容値＝23 ストリームが、そのままこの方式のビデオサーバーの許容値となることが分かる。即ちこれがこの方式のビデオサーバーの最大のボトルネックとなる。図 3.17 はこの最大ストリーム数＝23 を実現するための各要素の必要十分な機器台数を参考までに図 3.16 から導き出して示したものである。

以上のようにサーバーCPU のボトルネックが最も重大であることが判明した。このことは予想外の結果であると同時に、一般の通説をくつがえすものである。何故ならビデオサーバーの性能問題を扱った論文のほとんどは通信路の容量の問題と、サーバーについてはその“ディスク性能”に焦点をあてていることからもそれが分かる（例えば［坂本-95］、［Federighi-94］、［Tobagi-93］等）。本実験は、現状の市販製品で実現されている技術、特にディスク、SCSI、CPU、通信ソフトウェアの技術を使用してビデオサーバーを構築した場合には、実は CPU 性能が最大のボトルネックとなるという新しい事実を導き出した。将来的にこれらの現状技術が変革して行った場合、例えば CPU 負荷を

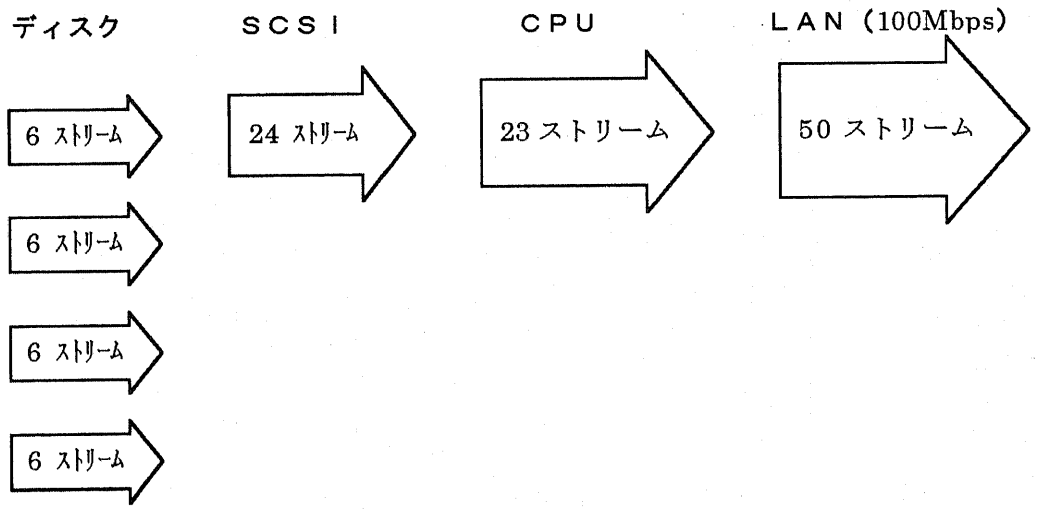


図 3.17　必要十分な構成

増大させている元凶である通信ソフトウェアが大幅に改良されるとか、CPU 性能がさらに飛躍的に増強される等の技術的変化が起きた場合には、この事実も変わってくるかもしれない。しかしながらそれらの改良は現在見えている訳ではなく、必ずしもあてにはできない。当然ながら本研究では現状の技術レベルに立脚した立場でビデオサーバーの高性能化に取り組むものであり、本実験結果を踏まえた形で先に進める。次の 3.3 節でこの CPU ボトルネックの解決法について考える。

３．３ 分散ＲＡＩＤ方式ビデオサーバー（ＲＡＩＤ０型）の提案と試作評価

３．３．１　ＲＡＩＤ適用の利点

当初ビデオサーバーの高性能化のためにはディスクが最も大きなボトルネックになると予測した。このための解決手段として RAID*ディスクシステムが有効であると考えられる。RAID が良いと考えた理由は以下の 3 点に集約出来る。

(1)Read Only

(2)均質なデータアクセス

(3)RAID4／5 のデータ訂正上の利点

RAID の一般的な利点は、多数のディスクに並列にアクセスが行われて高速化が図られることである。ビデオサーバーでも同様なことが言えるが、ビデオサーバーの場合にはこれに加えてさらに上記の 3 つの利点がある。以下にこの説明を行う。

(1) ビデオサーバーはほとんど Read Only に近い形態で動作する。勿論サーバーに新規のビデオプログラムを入れたり古いビデオプログラムを削除したりという Write 操作も存在するが、Read に比べて明らかにその比率は低く、又それは深夜等の利用率の低い時間帯を利用して行うことが可能である。このようにほとんど Read だけで“Write”が希にしか行われないため、“Write 性能が悪い”という RAID の最大の欠点を回避出来る。即ち RAID の“ええとこ取り”が出来るという大きな利点がある。

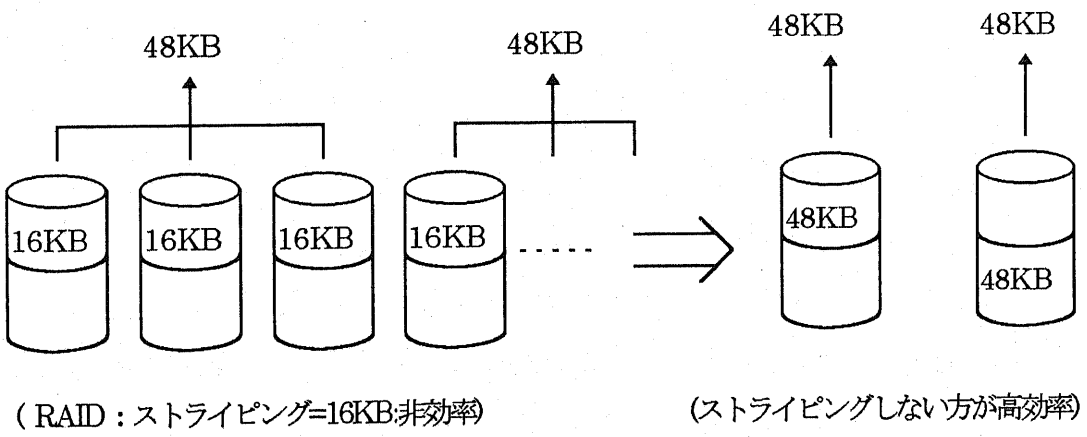


図 3.18　RAID が非効率な例

---

* 注) RAID については後述の 4.2.1 で説明される。

(2) ビデオサーバーではビデオデータを読み出すという均質なアプリケーションだけが走ることから、RAID を使った場合、次のようなメリットがある。RAID ディスクシステムで最も悩むのはストライピングのデータブロックサイズの設定法である。図 3.18 はこれを説明したもので、例えばブロックサイズを 16KB でストライピングしている時にアプリケーションが 48KB の単位でディスクアクセスすると、却って性能が落ちることを示している。何故なら 48KB 程度の少量なデータは 1 回の Seek 操作で読み取るべきであり、ストライピングしているために 3 台のディスクに Seek をかけて 16KB ずつ読ませるのは明らかに非効率である。このため通常 RAID を使う場合には、この上で動くいろいろなプログラムのディスクアクセス時のブロックサイズと、その使用頻度を調べて最大公約数的なブロックサイズを設定するが、その中で一部非効率なアクセスが起こることは避けられず性能低下の原因となる。これに対しビデオサーバーでは一定のブロックサイズでコンスタントにアクセスすればよいため、それに合ったブロックサイズ設定をしておけば、上記のような非効率な問題が起こらないという大きな利点がある。

(3) ビデオサーバーでは RAID4 ないし RAID5 において 1 つのディスクが故障した場合のデータ訂正時の効率が良いという特徴がある。RAID4／RAID5 ディスクシステムでは 1 台のディスクが故障しても運転は正常に続行されるが、この縮退運転時にはディスクシステム全体としての性能が低下するという問題がある。何故なら故障ディスクの場所を読み出す時には、それが属するストライピンググループの他の全てのディスクのデータを読み出し、これを基に故障ディスクのデータを復元する。即ちストライピンググループに属するディスク数を n とすれば、故障したディスクを読む場合には 1 台のディスクから読むかわりに（n－1）台のディスクから読まなければならない。このためディスクシステム全体としての性能低下を招く。これに対しビデオデータは連続的にアクセスされるため故障したディスクのデータブロックの前後のデータブロック、即ち（n－1）台のディスクから読むデータは、パリティを除いて元々その前後に読むはずのデータである。このため上記のような無駄が起こらない。これは 1 台のディスクが故障しても性能低下なしで運転が続行できることを意味し、非常に大きなメリットと言える。（この問題については 4 章で詳述する。）

以上のようにビデオサーバーに RAID を適用すると、RAID の通常の利点に加えて上記のような 3 つの大きな利点が上積みされることを発見した。このためビデオサーバーと RAID は非常に親和性が高いと考えるに至った。

３．３．２　分散ＲＡＩＤ方式の考案

以上のようにディスク性能の観点からは、ビデオサーバーの動作特性上 RAID 方式が非常によく適合することが明らかになったが、一方 3.2 節の実験結果からディスクでなく

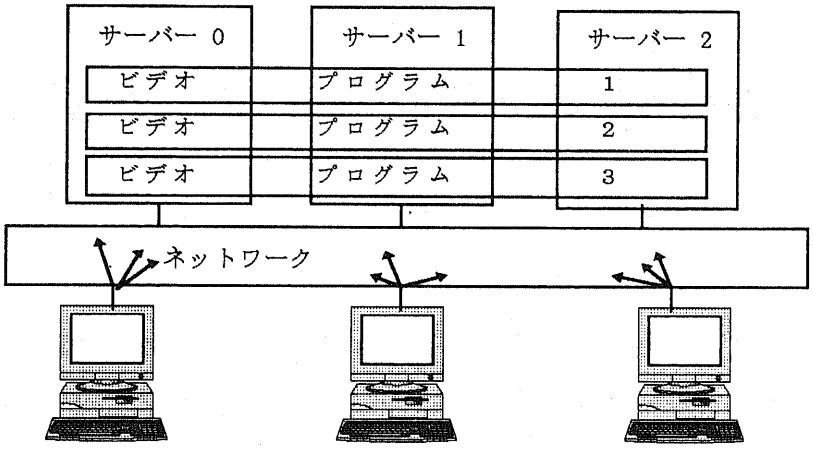


図 3.19　分散 RAID 方式ビデオサーバーの概念図

サーバーの CPU が最大のボトルネックであることが判明している。そこでこの RAID の利点を生かした形で何とか CPU ボトルネックを解決する方法はないものかと模索した。この結果 RAID のレベルを 1 段上げて、即ちディスクレベルでなくサーバーのレベルで RAID を構成する方法の考案に至った。図 3.19 はこの方法を概念図に示したものである。このように複数のサーバーを配置して、これをあたかも 1 つのビデオサーバーのように機能させようというものでる。図中の個々のサーバーは普及型の安価なパソコンサーバーを想定している。図 3.19 中各サーバーのディスク構成は明示されていないが、ディスクが 1 台でも複数台でもよく、又ディスクアレイ（RAID）でもよい。図のようにビデオプログラムは複数のサーバーにまたがって横断的にストライピングされて記憶される。ストライピングのブロックサイズはディスクの RAID と同様で 16KB～64KB 程度を想定する。通常の RAID

ではディスクに対してストライピングされるが、図 3.19 ではディスクを含んだサーバー全体を 1 つの記憶装置とみて、これに対してストライピングを行っていることになる。複数のサーバーはそれぞれ LAN 等のネットワークに接続し、それを介してクライアントにつながる。クライアント上のアプリケーションプログラムが各サーバーに順次アクセスしてデータブロックを読み出し、クライアント上に動画が表示される。このように複数台のサーバーが一体となって一つのビデオサーバー機能を実現する。これは、1 台のサーバーに RAID ディスクをつけた通常のビデオサーバーと比べると、その各ディスクないしディスクグループのところに CPU を付加したものと見ることが出来る。この方式によれば、サーバー数を n とすれば、1 個のサーバーCPU の負荷は 1／n に軽減されるため、ビデオサーバー全体の性能は n 倍に向上する。これにより 3.2 の評価実験で判明した“サーバーCPU が最大のボトルネック”という問題点が解決される。勿論 CPU のみに止まらず 3.2.1 で列挙したビデオサーバーの性能決定要素(1)～(4)において、(4)の LAN を除いた、(1)～(3)の性能が n 倍されるという効果をもたらす。

筆者等はこの方式を分散 RAID 方式と名づけた。分散 RAID 方式とは通常ディスクに対して使われる RAID の手法を、サーバにまで拡張して適用してデータの分散配置（ストライピング）により性能向上を図るものと言うことができる。従来、ユーザ数が増えて 1 台のサーバのビデオ供給能力を超えた時、図 3.20 のようにサーバ台数を増やして強化を図る方法が一般的によく用いられる。

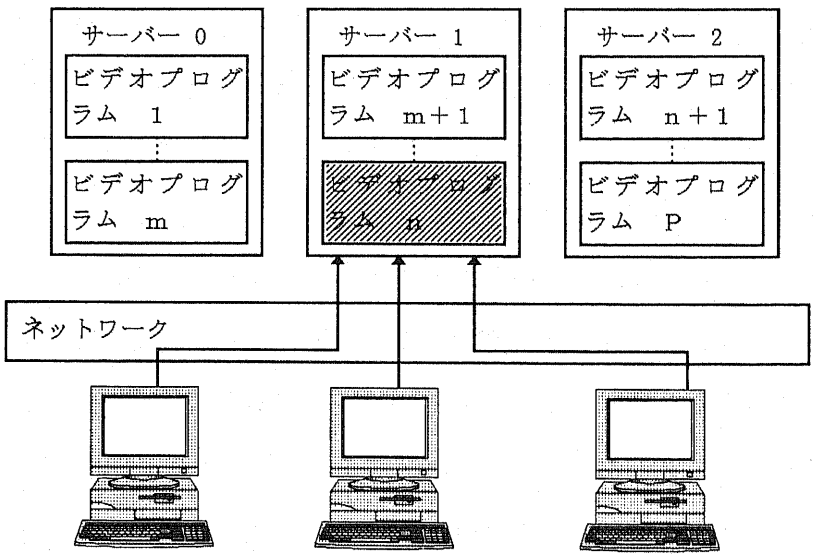


図 3.20　一般に行われるサーバー数増加による性能強化法

この場合データはビデオプログラムの単位でいずれかのサーバに分散配置される。即ちある１つのビデオプログラムはある１つのサーバーにしか存在しないため、図に示すように特定のビデオプログラムに要求が集中すると、結局１台のサーバだけが対応することになり、複数台化による性能向上効果が得られない。これに対し分散 RAID 方式では、１つのビデオプログラムが複数台のサーバにストライピングされて分散配置される。このため特定のプログラムにアクセスが集中しても複数のサーバが対応することになり、サーバ台数に比例したスケーラブルな性能向上が実現出来る。マルチプロセッサシステムにおいてプロセッサ数に比例した性能向上はなかなか難しいが、後述の実験で検証されるように、分散 RAID 方式ではほぼサーバ数に比例した性能向上が図られる。特に普及型の廉価なサーバを追加していくことによりリニアに性能向上が図れる点は大きな魅力である。また、ビデオデータにおいては各ストリーム毎に連続的にアクセスされ、各サーバに対するアクセス頻度が均等であるため、サーバ台数に比例したスケーラブルな性能向上が保証される。

３．３．３ 従来の研究

上記のように分散 RAID 方式は筆者等が独自に考案したものであったが、調査の結果カ

表 3.2　本研究と Swift の比較

項目	本研究	Ｓｗｉｆｔ
1サーバーに複数のディスクが付く方式の提案	○	×
クライアント主導の分散ストライピング方式の提案	○	× （分散ストライピングをとりまとめるサーバーを使用）
ビデオデータによる映像評価試験（目視、音声、コマ落ち率測定）	○	× （通常のデータの転送試験のみ）
実環境に近いデータ転送レートにて分散 RAID 方式の有効性を実証	○ （約 52Mbps）	× （約 8Mbps）
リニアな性能向上を実現	○ （サーバー数 1→3 台で、1→2.92 倍）	△ （サーバー数 1→3 台で、1→2.6 倍）

リフォルニア大学で Swift［Cabrera-91］［Long-94］という類似の研究が行われていることが判明した。Swift は高性能かつ高信頼な分散ファイルシステムを構築することを目指した研究であり、筆者等のビデオサーバー向けの応用を指向した研究とは方向を異にする。しかしながら分散 RAID 方式の基本的なアイデアの部分では本研究で提案するものと同等のものであることが判明した。このように残念ながら分散 RAID 方式の基本アイデアは既に存在するものであったが、本研究はビデオサーバーに的をしぼっているため、その実現方式の展開から試作評価にいたる課程で、表 3.2 に示すように多くの点で新規性を発揮することが出来た。

即ち本研究では、サーバー当たりのディスク台数を複数化し後述の図 3.22 のような新しいアレイ構成法を試みたこと、クライアント側のみにて分散ＲＡＩＤを実現する新手法（後述の図 3.25）を実現し評価したこと、Swift では通常のデータの転送性能の測定だけで映像データ試験は行われていないが、我々の試作では実際のビデオデータを流して正常な映像が得られることを確認し、コマ落ち率測定を行うなどビデオサーバーとしての各種評価を実施し、それへの適用の有効性を初めて明らかにした点、又 Swift では実験室レベルのデータ転送レート（最大約 8Mbps）の測定までしか行えていないが、本試作では実用レベルに近いデータ転送レート（最大約 52Mbps）まで実測しその実用に向けた有効性を検証したこと、又 Swift ではサーバー数に比例した性能向上が観測されていないが、我々の試作では概ねサーバー数に比例した性能向上を達成し本アーキテクチャの利点を一層明確に出来たこと等の新しい研究成果が得られた。

３．３．４ 分散ＲＡＩＤ方式ビデオサーバー試作の目的

3.3.2 で述べた分散 RAID ビデオサーバーの原理によれば、サーバー数を増加していくことによりリニアに性能が向上するスケーラビリティを備えたビデオサーバーを作ることが可能と思われる。しかしながらこの種の理論には落とし穴があることがつきものであり、試作による検証が不可欠である。例えば 3.2 で行った実験において、LAN 上のデータ転送量はサーバーからクライアントに向かうものが大半を占める。なぜなら LAN 上では 0.5KB 単位の転送が１つのトランザクションとして何度もくり返されるが、このトランザクション数で言えばサーバーからクライアント方向が圧倒的に多い。3.2 の実験ではこのデータ転

送のイニシエーター（送信側）は１つ、即ち１個のサーバーであった。即ちこれら大多数のトランザクションは１つのサーバーから発信されるため、それらが互いに衝突することはありえなかった。これに対し分散 RAID 方式では複数のサーバーが同時並行的に多数のトランザクションを発信することになるため、異なるサーバーの間でデータ転送の衝突が頻発することが予想される。この衝突による損失により目的とする性能が達成されない心配が大いにある。又本研究では特定の応用向けシステムの他にも、一般のオフィス LAN で使われるビデオサーバーを指向しており、一般のオフィスで通常使われている主流市販製品を使ったシステム構築を考えている。これはゼロからすべてを開発するのはシステムの開発費が膨大になるため事実上不可能であること、又製品化した場合のシステムの維持／グレードアップ費用がさらに継続的に必要になるという点からも、ある意味で当然の選択と言える。即ち市販製品を流用すれば、だまっていてもハード／ソフト共上方互換性を維持しつつ、その性能、機能共に向上していってくれるためシステムのグレードアップ費用が節約できるためである。さてこのような戦略をとる場合、通信用ソフトウェア、ファイル管理ソフトウェア、タスクスケジューラ等々のソフトウェアはすべて市販品を使用することとなり、こちらの思い通りに動くものとはならない。しかしながらこれらのソフトは世界的レベルでデファクトスタンダード化する傾向にあり、それらを使いこなしていく技術は今後益々重要となってくるため、時流に乗った手法であるとは言える。以上のようなフレームワークを前提とした場合、これらのソフトウェアは基本的にブラックボックスであり、それらの挙動については概略を予想できるにすぎない。このため実物を作ってみてその通り動くかどうかを確かめる作業の重要性が増す。

以上に述べた 2 つの点はほんの一例であり、この他にも種々の阻害要因が考えられる。即ち 3.3.2 で述べた分散 RAID ビデオサーバーがほんとうに理論通りに動作するのか否かについては、実際に作ってみて評価してみなければ分からない面が多々存在し、その理論を実機により検証することは不可欠である。

３．３．５ 分散ＲＡＩＤ０型のアレイ構成法

ここで試作を行うのは 3.3.2 に示した内容の分散 RAID ビデオサーバーであり、これは通常のディスクアレイで言えばいわゆる RAID0 型と言われるものに対応する。このため以下ではこれを「分散 RAID0 型」と呼ぶこととする。（これは 4 章で言及する「分散 RAID4 型」や［分散 RAID5 型］と区別するのに必要である。）分散 RAID0 型のアレイ構成法（ストライピング手法）を考える場合、１つのサーバーにディスクが１台しかつかない場合は、3.3.2 の図 3.19 “分散 RAID の概念図” から自明に一意的に導きだされる。しかしサーバーにディスクが１台というのは、3.2.7 の図 3.17 の結論からも自明なごとくシステムバランス上大いに非効率である。このためもし図 3.17 と同じ条件であれば、１つのサーバーに対してディスク 3～4 台構成とし最適化を図るのが適切である。１つのサーバーに複数台のディスクが接続する場合にはアレイ構成法として何通りかの方法が考えられるが、最も自然に発想されるのは図 3.21 と図 3.22 に示す方法である。

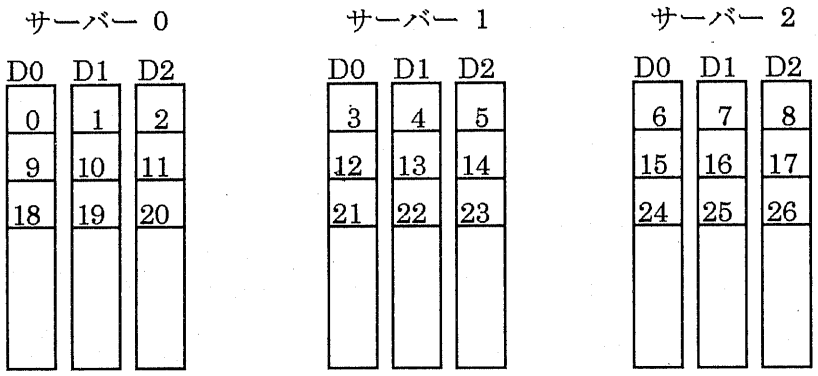


図 3.21　分散 RAID のアレイ構成ーその１

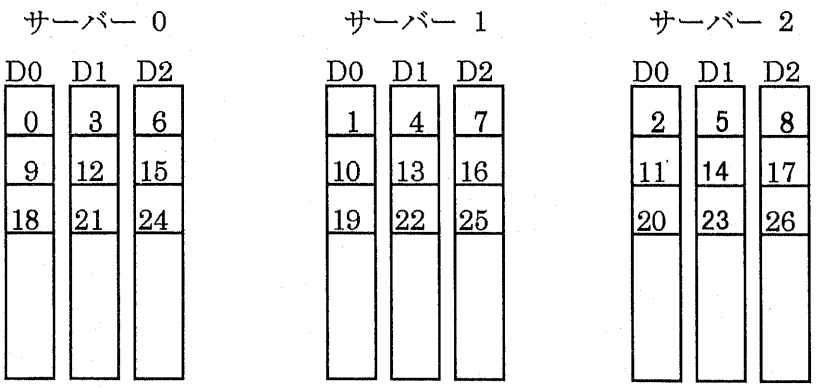


図 3.22　分散 RAID のアレイ構成ーその２　(採用)

これらは共に分散 RAID を構成するサーバー数が 3 台で、それぞれのサーバーにディスクが 3 台の構成の場合を示している。D0、D1、D2 は１つのサーバーに接続されるディスクを表しており、この中にデータブロックの並びが示されている。データブロックのサイズは現状のビデオ表示プログラムのアクセス特性に照らして 16KB～64KB 程度を想定している。図中各ブロックには番号が付されているが、これは１つのビデオプログラムがブロッキングされてこの順で各ディスクにストアされていることを示す。図 3.21 の“分散 RAID0 のアレイ構成－その１”の方法はディスクのレベルで次々と順次移行していくストライピング法となっている。これに対し図 3.22 の“その２”の方法はアクセスするデータブロックが毎回次のサーバーに移行していきサーバーレベルで 3 周すると丁度 9 台のディスクに一巡するストライピング法となっている。分散 RAID0 の場合には、後述の分散 RAID4 等と違い、これらのいずれを選んでも実質的な有為差は存在しない。サーバーのＯＳである Windows NT のソフトウェアストライピング機能との親和性がよい理由から、試作機には図 3.22 に示される“その２”のアレイ構成を採用することとした。

３．３．６ 分散ＲＡＩＤ０の実現方式

さてアレイ構成が決まった次は、いかなる方法でこのようなシステムを作り上げるかという問題である。まずはどのような部分にシステム上の改変が必要かということを、図 3.19 の分散 RAID の概念図を基にして考えてみる。図 3.19 はビデオデータ配信時の動きを示しているが、よく見るとサーバー側は特別な動きはしていないことが分かる。即ちクライアントからの要求に応えて指定されたデータブロックを転送すればよい訳で、これは通常のサーバーの動作となんら変わらない。これに対しクライアント側は、通常は盲目的にサーバーに対し、次のデータをくれ、又その次のデータをくれと言い続ければよかったものが、今度は大分大変になる。即ち今度はどのサーバーに対しどのデータブロックを要求するかということをクライアントは常に把握しそれを順次実行していかなければならないことになる。以上のようにビデオの配信／再生操作だけを見れば、分散ＲＡＩＤの機能はクライアント側だけで実現出来そうである。一方初期のビデオデータの登録機能を考えた場合には、サーバー側にも分散 RAID のストライピング方式に沿って、サーバーのディスクにビ

デオデータをロードしていく機能が必要となる。但しこの機能については、例えば複数のビデオを同時並行的に高速ロードするというような需要は今のところ存在せず、高速化の必要性が見当たらない。新着ビデオは１本２本という単位で入着し、それこそ夜中にでも１本２本とシーケンシャルに登録してゆくのが普通となっている。本研究で着目するのは多数のクライアントからのビデオデータ転送要求をタイムリーに実時間的にさばいてゆくビデオサーバーの性能、即ち配信／再生時の性能についてである。即ちビデオの登録機能については本試作で実験評価したい趣旨とははずれるため、これについては別プログラムを作成してその機能だけを実現することとした。

以上の考察から、サーバー側は通常市販されているものに特に変更を加えることはせず、上記の別プログラムによりビデオデータをストライピングしてロードすることのみを行う方向とした。そしてクライアント側にこの分散 RAID0 ビデオサーバーにアクセスする機能のすべてを持たせる方針とした。

分散 RAID0 のシステムを構成するコンポーネントについては、オフィス LAN で使用されるビデオサーバーが第一の目標であるので、この分野での主流製品を使用することとした。即ち図 3.19 を参照しながら見てみると、サーバーとしては Windows NT を、クライアントとしては Windows 3.1 を、ビデオ再生用のアプリケーションプログラムとしては Video for Windows を使用することとした。そしてネットワークとしては現在の主流は 10Mbps-Ethernet の LAN であるので、その発展形として 100Mbps-Ethernet の LAN を用いることとした。

さてクライアントにて分散 RAID0 のデータアクセス機能をいかに実現するか、特に Video for Windows という市販のプログラムを流用しながら、これを行うかが大きな問題となる。 Video for Windows に相当する部分を自主開発するのであれば、分散 RAID0 のアクセス機能をその中に組み込めばよいので問題とはならない。しかしこの部分を自主開発するのは開発量が膨大となるため事実上不可能である。この問題は MS-DOS の TSR（ Terminate and Stay Resident）機能を使うことにより解決出来る。この TSR 機能を利用することにより、クライアント上に分散 RAID0 のアクセス機能を実現出来る見通しがついた。

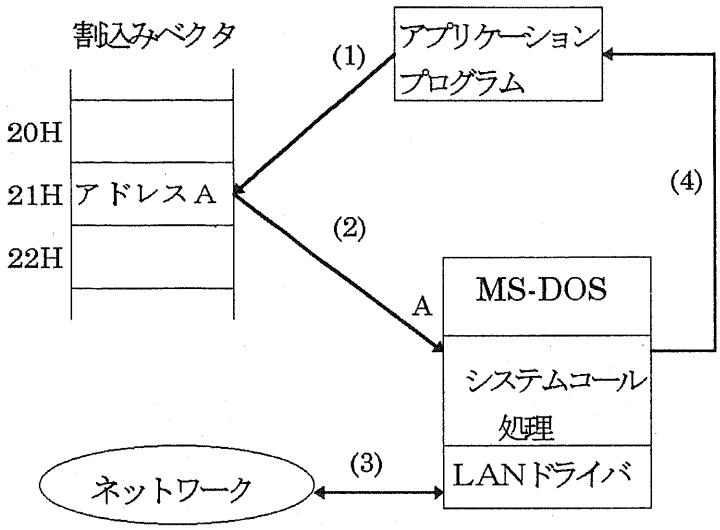


図3.23　通常のMS−DOSのシステムコール

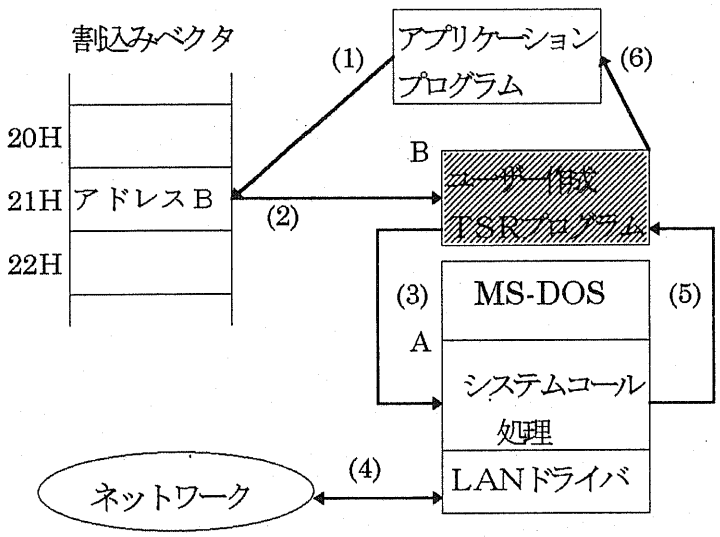


図3.24　TSRプログラムによるシステムコールのフック

TSRとはユーザー作成が可能なプログラムであり、MS-DOSの主メモリ上に常駐出来るプログラムである。そしてMS-DOSに対するシステムコールをフックする機能を埋め込ませることが出来る。フックとはアプリケーションプログラムからMS-DOSに対して出されたシステムコールを一時的に横取りする機能である。図3.23と図3.24はこのフックの動作を説明した図である。アプリケーションプログラムはファイル管理（例えばファイルのオープン）等の処理（時刻設定等も含む広範囲を包含）をOS（MS-DOS）に依頼する時にINT21というシステムコールを発行する。図3.23はこの時の通常の流れを示している。通

常割り込みベクターのベクターアドレス 21 には、OS のシステムコール処理ルーチンの先頭アドレスである“アドレスＡ”が設定されている。このためシステムコール INT21 が発行されると、割り込みベクタ 21 を介して、OS のシステムコール処理ルーチンに制御が渡り、OS による必要な処理が完了した後再びアプリケーションプログラムに制御が戻される。図 3.24 はこれに対し TSR によりフックされる動作が示されている。この場合には TSR の初期化ルーチンにより、割り込みベクタ 21 にはユーザー作成の TSR プログラムの先頭アドレスである“アドレス B”が初期設定されている。このためアプリケーションプログラムがシステムコール INT21 を発行すると図中斜線で示したユーザー作成の TSR プログラムに制御が渡る。ユーザー作成の TSR プログラムはファンクションコードを見て、例えば時刻設定のファンクション等であればこれを素通して OS のシステムコール処理ルーチンに渡す。もしこれが目当てのネットワークファイル READ のファンクションであった時には、これをつかまえそのシステムコールの内容の書き換えを行い、その後に OS のシステムコール処理ルーチンに渡す。リターン時にはその終了報告の内容を再び書き換えた後にアプリケーションプログラムに制御を戻す。このようにしてフック機能が実現される。アプリケーションプログラムには以上のことは全く見えないため、アプリケーションプログラムの変更が全く不要であることが重要なポイントとなる。

図 3.25 は以上に述べた TSR 機能を用いて、分散 RAID0 型ビデオサーバーを実現した方式を示す図である。図において下に書かれている３つの箱が３台のサーバーである。各サーバーの中には論理的な１台のディスクが書かれており、その中に１つのビデオプログラムに対するストライピングされたデータブロック番号が記入されている。真ん中に位置するネットワークを介してその上に書かれているのが１台のクライアントである。クライアントの中には３つのプログラムがあり、上には Video for Windows プログラムが、下には MS-DOS とその LAN ドライバーが一緒に書かれている。そして真ん中の斜線で示されたプログラム、“Striping Driver”と命名されたものがここで作成した TSR のプログラムである。この“Striping Driver”により分散 RAID0 型のビデオサーバーが実現される。

“Striping Driver”は次のように動作する。 Video for Windows（アプリケーションプログラム）が論理的なドライブ D の D:～.avi ファイル（～はそのビデオの固有名）のブロック 0 に対してシステムコール INT21 を使って READ を発行する。これが完了すると次に

同じくブロック 1 に対して READ を発行する。同様にしてブロック 2、3、4…と順次ビデオデータを読み取って映像を放映する。この時ストライピングドライバ (TSR プログラム) は次のようにシステムコールの書き換えを行う。即ちD：～.avi ファイルのブロック 0 の READ に対しては E:～.avi のブロック 0 の READ に、 D：～.avi ファイルのブロック 1 の READ に対しては F:～.avi ファイルのブロック 1 に、 D：～.avi ファイルのブロック 2 の READ に対しては G:～.avi のブロック 2 への READ に、 D：～.avi ファイルのブロック 3 の READ に対しては E:～.avi のブロック 3 の READ に、・・というように置き換えていく。これによりアプリケーションプログラムが論理的な 1 つのファイルD：～.avi から順次データを読んでいくと、実際にはこれが E:～.avi 、F:～.avi、 G:～.avi という 3 つのファイルから交互に順次読み取られていくことになる。

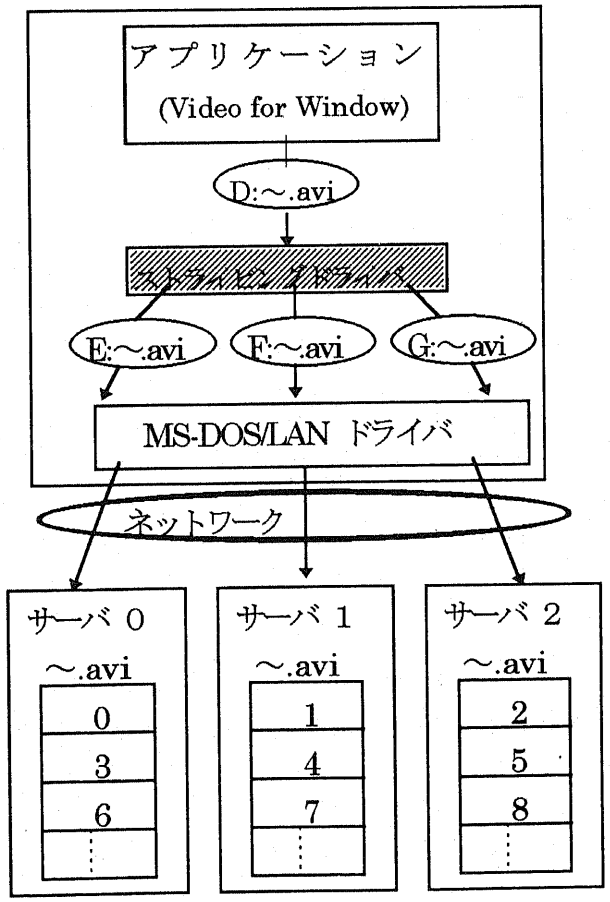


図 3.25　RAID0型の実現方式

図 3.25 に示されるようにこれらの 3 つのファイルはそれぞれ 3 つのサーバーに分散して配置されている。そのため順番に各サーバーを移動しながらデータが読み取られていくことになり、3 台のサーバにストライピングされたビデオデータが順次読み出されていくことになる。以上のようにして、TSR プログラムである “Striping Driver” により分散 RAID0 方式ビデオサーバーの機能が実現される。この “Striping Driver” プログラムは C 言語により作成し、ボリュームは 848 ステップとなった。

3．3．7 試作機のシステム構成

図 3.26 はこのシステムを実装する試作機のシステムの構成を示す図である。

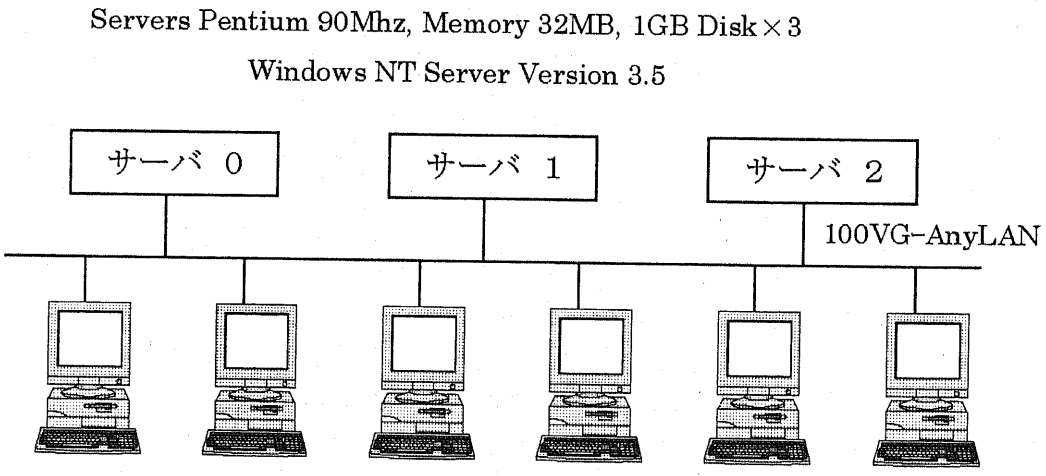


図 3.26 分散 RAID0 ビデオサーバー試作機のシステム構成

前述のように現状の主流ソフトウェアの上で動かすという方針であるため、サーバとしてはWindowsNT3.5を搭載した、ごく普通のパソコンサーバー3台を用意した。各サーバには1GBクラスのディスクを3台接続し（ディスクは合計で9台）、Windows NTのソフトウェアストライピング機能を設定した。このストライピングのデータブロックサイズは16KBに設定した。クライアントにはこれまたWindows 3.1（この下にMS-DOSが走る）が載ったごく普通のパソコンを使用し、これに前述の“Striping Driver”を搭載したものを合計で6台用意した。 LANは100Mbps-Ethernet（100VG-AnyLAN）1本を使用し、上記の3台のサーバーと6台のクライアントをつなげた。

３．３．８ 性能評価の設定条件

以上のようなシステム構成が設定されて次にいよいよ性能評価のための試験に入る。この試験に適用した各種の設定条件について以下に記述する。まずビデオ再生用ソフトウェアとしては米マイクロソフト社のVideo for Windowsプログラムを使用した。次にビデオデータは3.2の実験と同じもの、即ちVideo for Windowsプログラムに付属して付いてきたAVI形式の、1.2Mbpsのレートの約5分間の長さのビデオデータを使用した。このビデオデータの6本分のコピーを分散RAID0ビデオサーバーのディスク（3台のサーバーの合計9台のディスク）上にロードした。このローディングは前述のように、別途作成した別プログラムにより行った。ここで6本分のコピーというのはクライアント数の6台と対応しており、各クライアントは常にそれ専用の1本のビデオプログラムにアクセスするようにした。このように1つのクライアントが常に同じビデオプログラムにアクセスすると一見サーバー側ディスクのSeekのランダム性を減少させるようにも思われるが、6台のクライアントからのデータ要求はごちゃ混ぜに交じり合ってサーバー側に到着するためその心配はない。

さてクライアントの数は6台揃えるのが限度であった。これでは高性能ビデオサーバーの試験、即ち多数のクライアントに同時ビデオ配信出来ることを確認する試験は行えない。このため3.2の実験で使用したものと同様の擬似クライアントプログラムを使用することとした。即ちビデオを放映したりコマ落ち率の測定を行うのは1台のクライアントのみとし、他の5台のクライアントでは疑似クライアントプログラムを走らせる。擬似クライア

ントプログラムはVideo for Windowsが載っている場合と同様にビデオサーバーに対してデータ要求を行うが、これを可能な限り目一杯行う仕様となっている。このため1つのクライアントをあたかも多数のクライアントであるかのように振る舞わせることが出来る。ここで若干心配されるのは、擬似クライアントプログラムが動く5台のクライアントでは、映像／音声の目視チェックやコマ落ち率の測定が出来ないことが問題とならないかという点である。これは以下の理由により問題とはならない。即ちこの評価において注目する点は、多数のビデオデータ要求のトランザクションが錯綜し、これらをサーバーのCPUやディスクそしてLANがいかに手際よくさばくかという所にある。当然トランザクション数が増えて混み合ってきた時が焦点となる。即ちCPU、ディスク、LANに対する負荷が増大した状況で、CPU、ディスク、LANがこれらのタイムクリティカルなデータ要求をオンタイムで処理できるか否かである。この時に処理される錯綜した多数のトランザクションの中で、Video for Windowsにより実際に放映している1台のクライアントからのトランザクションであるという区別は当然つかない。即ちこの1台のクライアントからのデータ要求は他の5台のクライアントからのデータ要求と対等な立場で渾然一体となって処理される。即ちこの1台の実クライアントは他の多くの擬似クライアントとこの意味では対等である。故に実際に放映しているこの1台のクライアントを標本として抽出しても偏りは生じない。それ故この1台のクライアントのみにおいて測定したコマ落ち率や目視チェックの結果は全体を代表していると見ることが出来る。

最後にこの評価試験においてストリーム数を上昇させていく方法について述べる。3.2の評価実験とは異なり、今度は分散RAID0の高性能ビデオサーバーであるから発生するストリーム数も格段に多い。ストリーム数を上昇させるのは単純にクライアントを増加していく方法により行った。即ち最初に映像を放映する1台のクライアントのみ動作させて測定を行い、次に擬似クライアントプログラムが動くクライアントも1台動作させて測定を行い、次に擬似クライアントプログラムが動くクライアントをもうに1台動作させて測定を行い、・・というようにしてストリーム数を増やしていく。これは擬似クライアントプログラムは何本の擬似クライアントを走らせるというような設定機能がなく、常に目一杯のデータ要求を行う仕様となっていることによる。

３．３．９ 性能評価結果とその考察

以上の設定条件に従って分散 RAID0 型ビデオサーバー試作機の性能評価試験を行った。サーバーの数は 3 台あるが、これを 1 台だけ、2 台だけ、3 台というように動作させていき、それぞれについて性能測定を行った。（サーバー数が 1 台の場合は分散 RAID0 とは言えなくなるが、見方を変えれば分散 RAID0 の特殊なケースと言えなくもない。）この測定結果全体をまとめたものが表 3.3 である。サーバーが 3 台の場合のコマ落ち率についてグラフ化したものが図 3.27 である。図 3.28 はサーバ数がそれぞれ 1、2、3 台の 3 つのケースで、クライアント側に供給される総ストリーム数に対してサーバ側の CPU 使用率を測定したものである。（内容は表 3.3 と同じ）各ケースにおいて最初のストリーム数＝1 の所の点は、映像を放映する 1 台のクライアントだけが動作した場合である。その次の点は疑似プログラムが動くもう 1 つのクライアントが動作した時であり、同様にしてクライアント数を増やしていったものである。

以下にいくつかの項目に分けてこの測定結果に対する考察を行う。

［1］ストリーム数の増加

図 3.28 のグラフないしは表 3.3 の中のサーバーが 1 台のところに注目してみる。各測定点はクライアントを 1 台 2 台・・と順に増やしていったところのデータである。最初は実放映するクライアントを 1 台だけ動かした状態なので、ストリーム数は 1 である。次に擬似クライアントプログラムが走るクライアントを 1 台追加するとストリーム数は 10.25 に増える。即ちこの追加したクライアント 1 台で 9.25 個の擬似クライアントを生成していることになる。このように擬似クライアント数が小数点表示になるのは、3.3.8 の終わりで述べたように、擬似クライアントプログラムは常に目一杯のデータ要求を行う仕様になっているためである。次にクライアントを 1 台追加すると全体のストリーム数は 16.65 となる。即ちこの 2 番目に追加したクライアントは 6.4 擬似クライアントを生成していることになる。同様にして 3 番目に追加したクライアントは 2.84 擬似クライアントを、4 番目に追加したクライアントは 0.68 擬似クライアントを生成していることになる。即ちストリーム数の増分は 9.25 ストリーム、6.4 ストリーム、2.84 ストリーム、0.68 ストリームと減少していく。このようにリニアに増加しない原因は、サーバ CPU のビジー率が増えると、クライ

アント側の要求が待たされることが増えて、疑似プログラム・クライアントからの読み出しデータ量が減少するためと考えられる。

表 3.3　分散 RAID0 試作機の評価結果

サーバー数	ビデオストリーム数	ＬＡＮデータ転送レート	ＣＰＵ使用率	コマ落ち数	コマ落ち率
1	1.0	1.2	5%	0	0.00%
	10.25	12.3	46%	0	0.00%
	16.65	20.0	80%	0	0.00%
	19.49	23.4	96%	0	0.00%
	20.17	24.2	99.5%	0	0.00%
2	1.0	1.2	3%	0	0.00%
	10.4	12.5	20%	0	0.00%
	19.4	23.8	44%	0	0.00%
	26.9	32.3	61%	0	0.00%
	32.9	39.5	78%	0	0.00%
	36.7	44.0	88%	2	0.04%
3	1.0	1.2	2%	0	0.00%
	10.9	13.1	12%	1	0.02%
	19.8	23.8	28%	0	0.00%
	28.6	34.3	42%	0	0.00%
	36.4	43.7	56%	0	0.00%
	42.8	51.4	66%	1	0.02%

［2］コマ落ち率

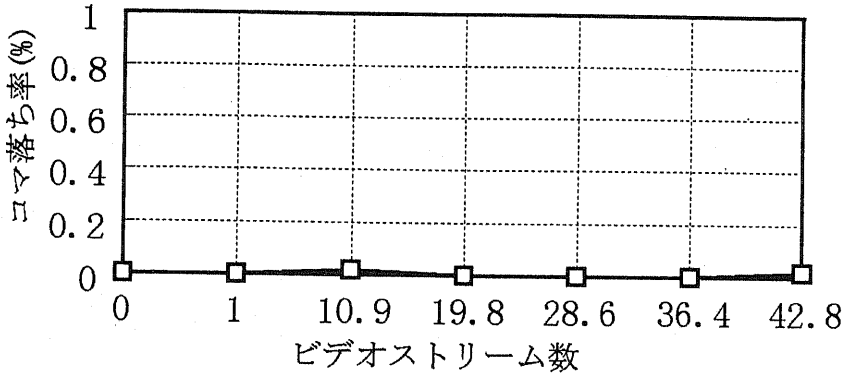


図 3.27 コマ落ち率(%) (サーバー数＝３の場合)

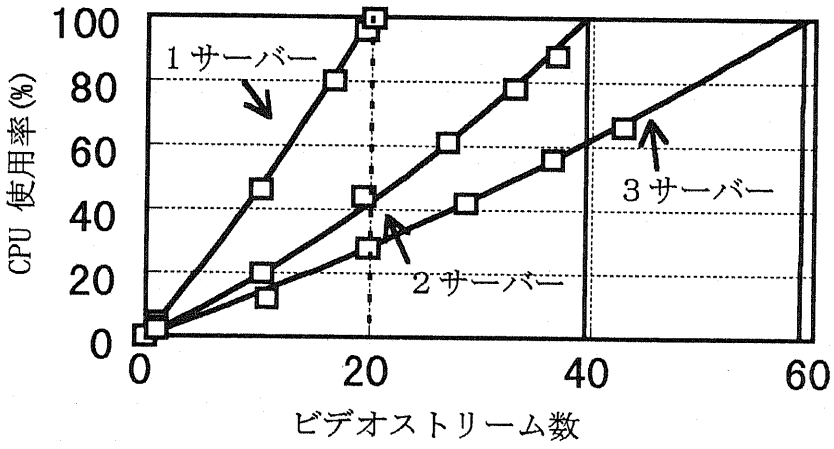


図 3.28 サーバーＣＰＵ使用率

表 3.3 のコマ落ち率の項は、放映クライアントでのコマ落ち率を示したものである。コマ落ち率が 1%以上になるとはっきりコマ落ちに気付くが、表 3.3 ではほとんど 0%であり、多いところでも 0.04%となっている。即ち表 3.3 中のコマ落ち率の測定結果はきわめて良好な値を示していることが分かる。これから１～３サーバの全てに渡って非常に良好な映像が供給出来ていることが確認できた。

なおサーバ数＝1 のケースで CPU 使用率＝100%の所を注目すると、ここでもコマ落ち

率＝0%となっている。即ちサーバの CPU ビジー率が 100%であっても、クライアント側にコマ落ち率 0%の良好な映像が転送出来るという注目すべき結果も得られた。

［3］ＬＡＮ上の競合の問題

LAN を使ってデータ転送を行う主体（イニシエータ）は、回数、データ量共にクライアント側よりもサーバ側が圧倒的に多い。近似的にサーバだけが LAN のイニシエータであるとすると、1 サーバの時には LAN 上でデータ転送がぶつかることはないが、2 サーバ、3 サーバの時はサーバ間でぶつかりが起き、片方に待ち時間が出来るため性能低下を来す可能性がある。この条件が最も厳しいのは 3 サーバで 6 クライアント（42.8 ストリーム）の場合であるが、表 3.3 中のそのコマ落ち率は 0.02%と良好である。即ち、少なくともこの時点までは上記による性能低下は起きていないことが確認出来た。

［4］分散ＲＡＩＤ方式の効果

さて 3.2 節の結果から、サーバーの CPU 性能が最大のボトルネックであることが分かっている。（注：この試作システム自体では必ずしもそうでないかもしれない。ここでもしディスクが最大のボトルネックであるなら、各サーバーのディスク台数を増やしていくことが出来る。このようにして最後に CPU ボトルネックが残る。）即ちこの方式のビデオサーバーシステムではサーバーの CPU 性能がシステム全体の性能を決定することになる。図 3.28 はこの側面からみたグラフ、即ち横軸の供給ビデオストリーム数に対しサーバーの CPU 使用率を示したグラフである。このグラフにおいてサーバの CPU 能力を 100%使い切った時のストリーム数が、そのビデオサーバの最大供給能力と見ることが出来る。図 3.28 からこの値を読みとると、サーバ数 1 台、2 台、3 台の時のビデオストリーム数供給能力は、それぞれ 20.2、40.7、59.0 ストリームとなっている。即ち 1 台、2 台、3 台の時の比率は 1：2.01：2.92 となっている。これがそれぞれの場合（サーバーが 1 台、2 台、3 台）のビデオサーバーの性能の比率であるが、、これはサーバ数の比 1：2：3 にきわめて近いものであることが分かる。即ち筆者らが分散 RAID0 方式のビデオサーバで目指したところの、サーバ数に比例した性能向上という目標が達成されていることが確認出来た。

３．３．１０ まとめ

分散 RAID0 方式ビデオサーバーの提案を行い、試作及び評価を行った。性能評価の結果、サーバー数に比例したリニアな性能（供給可能ビデオストリーム数）向上が図られることが確認出来た。（Swift［Cabrera-91］［Long-94］の例ではこのようにきれいに比例していない。）又この時の映像品質もコマ落ち率が 0～0.04%ときわめて良好なものが得られた。又複数のサーバがイニシエータとして 1 つの LAN を共有してデータ転送することにより生ずる競合による性能低下は、筆者等が観測した範囲（42 ストリーム）では（サーバー側において）起きていないことが確認出来た。又、Swift では分散 RAID において、通常のデータの転送性能を測定したにとどまっているが、筆者等は実際に動画を表示しながら確認し、かつコマ落ち率測定プログラムにより画面の品質の測定も並行して行った。このことから分散 RAID 方式をビデオサーバーに適用した場合の有効性が、初めて確認されたと言える。また、この時サーバの CPU の使用率が 100%になるまで良質の動画が供給出来るという興味深い評価結果も得られた。以上得られた試作評価結果は、分散 RAID 方式ビデオサーバーの実現性を大きく確実にするものと考えられる。

さて以上のようにビデオサーバーとして非常に有効な方式が示された訳であるが、次にこの欠点について考えてみる。第１に信頼性の低下の問題がある。この方式ではサーバー数がいくら増えてもそれに比例して CPU 性能が向上する。これと表裏一体の関係として、性能向上のためにサーバー数をどんどん増やして行った時に、サーバー数をｎとすれば、システム全体の信頼性は１／ｎに低下するという問題が存在する。この問題については次の第 4 章にてその解決策について考える。

さて第２の欠点として、クライアント側のプログラムに負担がかかるという問題がある。この場合クライアント CPU の負荷については、この方式をとることにより 10 数%増えるだけであり、3.2.1 にて述べたように、元々クライアントは１本のビデオを放映しているだけのためほとんど問題にとはならない。むしろこのタイプのビデオサーバーに対してはクライアント側に専用のアクセスプログラムが必要という点が、煩わしさを生ずるという点で問題になりそうなことが予想される。話は全く変わるが、鎌倉市では新しいゴミの回収方式を 1997 年 5 月から一部地域で試行、10 月より本試行を開始しテレビでも話題となっ

た。これによれば各家庭は従来より遥かに細かい分類でゴミを出さなければならない。例えば紙類でも何種類かに別けて出さねばならず、例えば牛乳パックは中を洗って開いて束ねて出さねばならない。もしも非常に優秀なゴミ分別機があってゴミを全部一緒に出しても自動的に全部きれいに分別してくれるならそれに越したことはない。こういうものがない状況で地球環境保護に取り組む方策として、住民側に一部負担を求めるという鎌倉市の例は１つの解決策を示している。そして現在ではこの方式が現実に施行されている。これと同様な視点で１台の万能で強力なサーバーがあって、これがなんでも全てやってくれるならばクライアントから見れば、楽であり言うことはない。しかしながらそのような１台で強力なビデオサーバーがない以上、サーバー側のCPU能力が不足するという問題に対処する手段として、クライアント側に１部負担を求めるというのも１つの解決策として十分に有り得るのではないだろうか。 即ち各クライアント毎に分散RAID0ドライバーを搭載するという手間が発生するが、これは許容される範囲ではないかと考えるものである。

３．４　結言

一般のオフィス LAN 等の一つの応用として用いられるような価格性能比にすぐれたビデオサーバーを目指して実験及び試作評価を行った。LAN 上で使われるビデオサーバーの性能を決定する要因として、サーバーのディスク、そのディスクが接続されるバス(SCSI)、サーバーCPU、LAN（10Mbps 及び 100Mbps-Ethernet）の評価を行った結果、前２者のボトルネックは検出されたが、簡単な改良と増設により解決可能なことを実験で確認した。又 LAN 性能によるボトルネックは特に高画質を実現する上で非常に重要であることが実験により確認されたが、この問題もＬＡＮ増設という回避手段があり、LAN を３本に増設した実験を行って実際にうまくいくことも確認した。これに対しサーバーの CPU 能力のボトルネックは深刻な問題であり、容易な解決手段がないことが分かった。

このサーバーCPU のボトルネックを解決する目的で、分散 RAID 方式のビデオサーバーを提案し、その試作評価を行った。分散 RAID 方式のサーバーの試作例としてはカリフォルニア大の Swift［Cabrera-91］［Long-94］がある。Swift は基本的に分散 RAID 方式のファイルサーバーであるため、通常のデータ転送性能についてのみ試験が行われている。このためビデオデータが途切れることなく送れるか等のビデオサーバーとしての評価は全く行われていない。又その実測データ転送レートは最大でも 8Mbps（ビット／秒）程度と、実験室レベルの規模に止まっている。これに対し本研究の試作機では実際のビデオストリームを転送し、各種の測定時に常に、クライアントで放映して画質と音声品質を確認しコマ落ち率測定を実施して、ビデオサーバー向け応用に対する有効性を初めて実証した。又データ転送レートも最大で 52Mbps（1.2Mbps のビデオストリームが 40 本以上。画質も良好。）を実測する等、初めて実用レベルに近い規模での検証を行った。又、スケーラビリティの問題については、Swift ではプロセッサ台数に比例した性能向上は図られていないのに対し（例えばプロセッサ 3 台で 2.6 倍の性能）、我々の試作機ではプロセッサ台数にほぼ比例した性能向上が図られ（例えばプロセッサ 3 台で 2.92 倍の性能）、分散 RAID 方式の利点をより鮮明にすることが出来た。以上から分散 RAID 方式ビデオサーバーのフィージビリティが大いに高まったものと考える。

第４章　高信頼型・分散ＲＡＩＤ方式ビデオサーバーの提案と試作

４．１　緒言

前章においてビデオサーバーの高性能化を図る手法として、分散 RAID0 型ビデオサーバーの提案を行いその試作結果について述べた。その結果この方式によれば、ほぼサーバー数に比例してビデオ配信性能が向上するビデオサーバーが実現出来ることが確かめられ、非常に好ましい結果が得られた。しかしながらこの方式には一つの欠点が存在する。それはサーバー数の増加に比例してビデオ配信性能が向上する反面、それに反比例してビデオサーバー全体の信頼性が低下するという問題である。本章では分散 RAID0 型ビデオサーバーのこの欠点を克服するために、分散 RAID0 型ビデオサーバーの改良について考え、その改良方式の提案と評価を行う。この信頼性に対する欠点が克服されて初めて、分散 RAID 方式ビデオサーバーの実用性（フィージビリティ）が確実なものとなる。

４．１．１ 分散ＲＡＩＤ０型ビデオサーバーの欠点

ここで筆者等が分散RAID方式と呼ぶものは、通常ディスク群に対して適用されるRAID手法を、サーバー群にまで拡張して適用を図ったものである。前章にて分散RAID0型ビデオサーバーの提案を行い試作を試みた。その結果サーバー数に比例してビデオ配信性能が向上するという、筆者等が当初意図したきわめて好ましい結果が得られた。

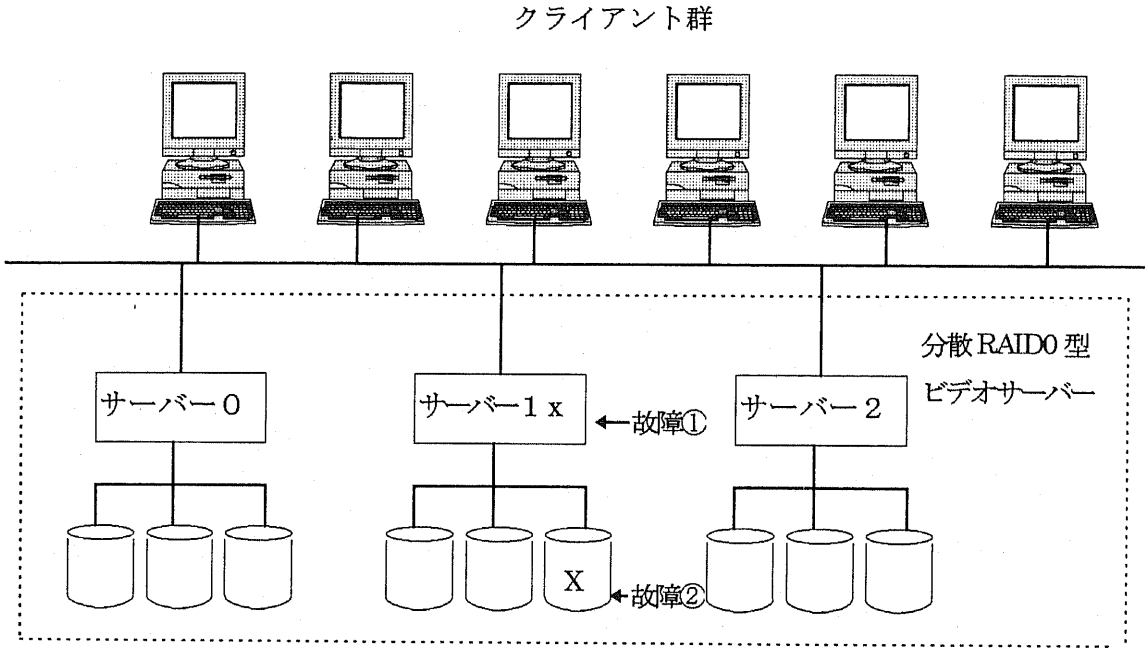


図 4.1　　分散 RAID0 型ビデオサーバーの欠点

しかしながら分散RAID0型ビデオサーバーには一つの欠点が存在する。これを図4.1に示す。分散RAID0型ビデオサーバーではその構成上、どれか一つのサーバー本体が故障しても（図4.1の故障①）、ないしはどれか一つのディスクが故障しても（図4.1の故障②）、正常に機能が実行出来なくなる。そしてサーバー数を増やせば増やす程コンポーネント数が増え、全体の故障率が増大する。単純化してサーバー本体だけで考えると、一つのサーバーの故障率をｆとすれば、ｎ台のサーバーから構成されるビデオサーバーの故障率はｎ・ｆとｎ倍になる。又、平均故障間隔MTBF（Mean Time Between Failure）で言えば、MTBFは1／ｎに減少する。このように分散RAID0型ビデオサーバーでは、サーバー数を増やしていって性能向上をすればする程、これに反比例して信頼性が低下するという問題

がある。これは実用化の観点から大きな問題である。本章ではこの問題の解決策について考える。

４．２　分散ＲＡＩＤ４型ビデオサーバーの提案と試作

４．２．１　分散ＲＡＩＤ４型ビデオサーバーの提案

分散 RAID 方式は、もともと RAID ディスクアレーのアナロジーから導出したものであるが、RAID ディスクアレーは本来高い信頼性を売り物にする。このことから RAID ディスクアレーの高信頼化手法が流用できないか検討する。

RAID とは D.A. Patterson 等［Patterson-88］により命名されたもので、Redundant Arrays of Inexpensive Disks の略語である。考え方としては［Salem-84］、［Salem-86］や IBM の特許等、それ以前から存在していた。［Patterson-88］では RAID を、RAID1～RAID5 の 5 種類に分類している。その後ストライピングのみで故障訂正機能のない RAID0 や、二重故障を訂正出来る RAID6 なども提唱され公の認知する所となっている。RAID1 はデータを重複して記憶するミラーディスク手法のことで、通常よく使われている。RAID2 はハミングコードにより故障訂正を行うものであるが、実用例はほとんどない。RAID3～RAID5 はパリティにより故障訂正を行うもので、グループ中の 1 台のディスクが故障してもデータが復元出来、処理が継続出来るようにしたものである。RAID3 はグループ中のディスクに対し一斉に読み／書きをするもので、初期には商用化例が多かった。これに対し RAID4 と RAID5 はディスクを 1 個ずつ読み／書きするもので、RAID5 は RAID4 の改良型である。このため最近の商用機では RAID といえば、RAID5 が多くなっている。

上記の RAID ディスクの高信頼化手法の中で、まずは RAID4 に注目し、この方式のアナロジーから改良策を考える。

［1］RAID4 ディスクの原理

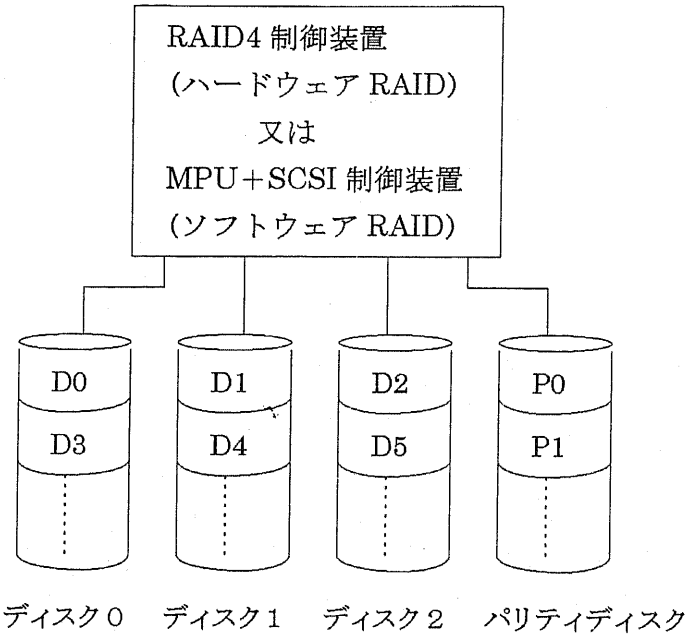


図 4.2　RAID4 ディスクアレイの原理

図 4.2 を使って RAID4 ディスクアレーの故障回復原理を説明する。ディスク群が RAID4 制御装置に接続され、それがサーバー等の本体に接続されるいわゆるハードウェア RAID と、複数のディスクが SCSI 制御装置に接続され、それがサーバー等の CPU に接続されるごく通常のディスク接続構成において、ソフトウェアにより RAID4 機能を実現する、いわゆるソフトウェア RAID の 2 つの実現方法があるが、原理的には両者とも同様である。図 4.2 の 4 台のディスクの中で一番右にパリティのみを記憶するパリティディスクが存在するのが RAID4 方式の特徴である。RAID0 と同様に、データは D0、D1、D2、D3、・・というブロック単位で、ディスク間をまたがって記憶される。この時ディスク 0 の D0、ディスク 1 の D1 ディスク 2 の D2 の間で、ビット対応に Exclusive OR 演算を行い、この結果（＝パリティ）をパリティディスクの P0 ブロックに記憶する。同様に D3、D4、D5 の Exclusive OR をとったものをパリティディスクの P1 ブロックに記憶するというようにして、ディスク全体に渡ってパリティを設定する。これにより、この中のある 1 つのディスク、例えばディスク 1 でディスククラッシュ等が発生しても、そのデータを復元することが可能となる。ディスク 1 が故障した場合、その中のデータブロック D1、D4、・・・が

読めなくなるが、これらはパリティディスクを使って次のように復元できる。

D1＝D0　xor　D2　xor　P0

D4＝D3　xor　D5　xor　P1

.

.

一般にRAID4ディスクアレイでは、いずれか1台のディスクが故障しても、その内容は復元され運転は続行される。但しこの時次のような欠点も存在する。プログラムが故障ディスクのデータを読みにいった場合、通常ならその1台のディスクから読めばいいものを、他の全てのディスク（図4.2では3台のディスク）から読み、これらをExclusive ORしてデータを復元するという操作が必要となる。そのため、この運転モード（縮退運転モードと呼ぶ）では性能が低下するという問題がある。

さらに通常運転モードでも、データのWrite時にパリティディスクを書き換える必要があるため、Write性能が悪くなるという問題もある。例えば図4.2でデータブロックD2を書く場合に、通常はディスク2にデータブロックD2を書くという1回の操作で済む。これに対しRAID4ディスクアレイのWrite操作では、まずディスク2とパリティディスクからD2とP0の現在のデータを読み出す。これら2つと、D2に書く新しいデータの間でExclusive OR演算を行い新しいパリティP0を求める。この後、新しいデータD2をディスク2に、新しいパリティP0をパリティディスクに書き込む。このようにWrite時にディスクアクセスが、通常は1回で済むところをRAID4では4回必要となる。さらにExclusive OR演算により新しいパリティの値を求める操作も必要となる。このためRAID4ディスクアレーは本質的にWrite性能が悪いという欠点を持っている。

さて故障復元の問題に戻って、RAID4ディスクでは何故このように失われたデータの復元が可能なのかを考察する。パリティは主メモリの故障検出等に使われるもので、故障検出能力しかないはずである。何故RAID4ではパリティを使って故障訂正（復元）まで出来るのだろうか。その答えは故障箇所が特定出来るという点にある。主メモリ等で8ビットのデータに対して1ビットのパリティを付加する場合には、パリティエラーが検出出来るだけである。この場合単に9ビット中のどれか1ビットが故障しているということが分かるだけである。これに対しディスク装置はそれ自体の中で、パリティチェックより遥かに

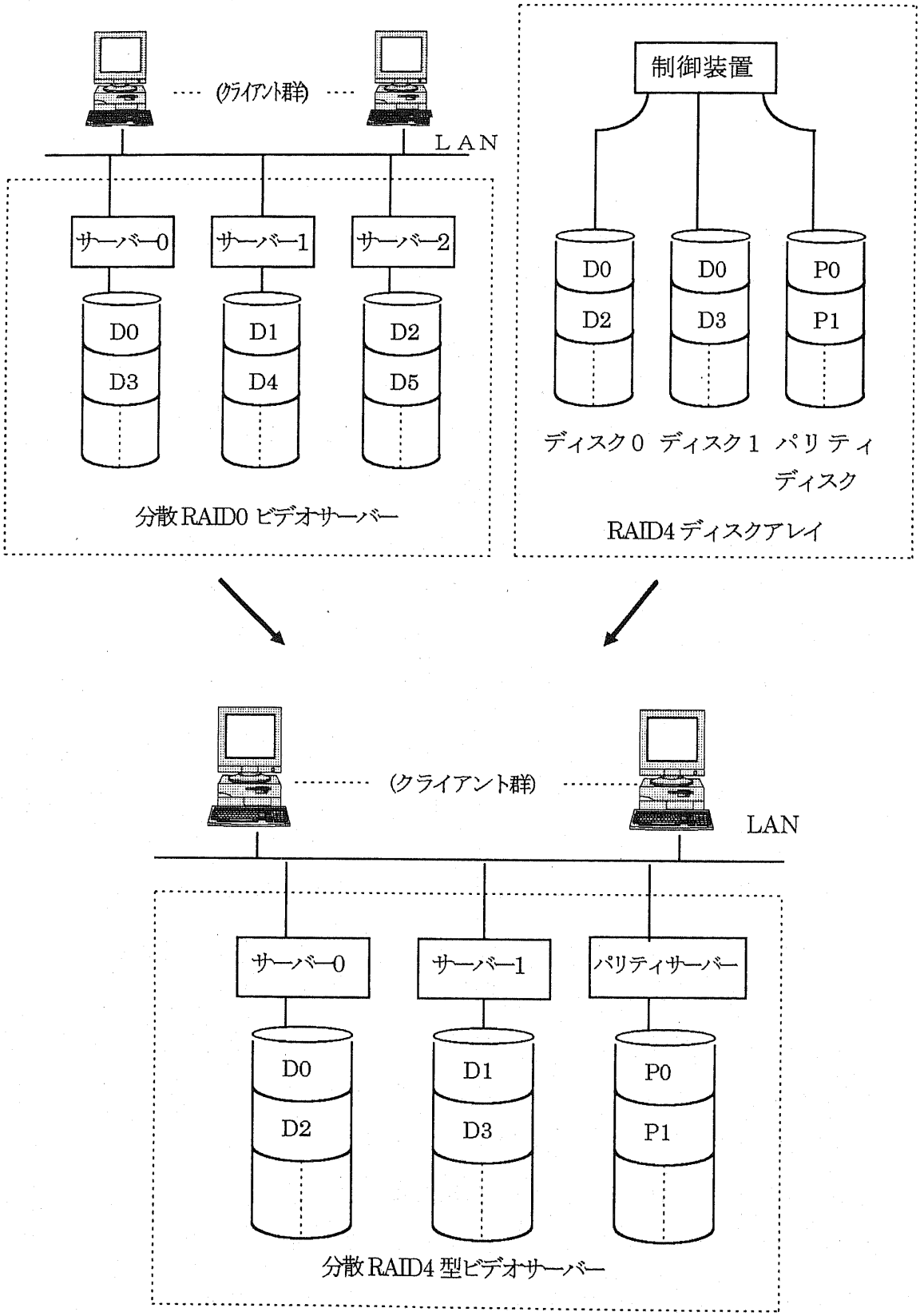


図4.3　分散RAID4方式ビデオサーバーの導出

強力な、昔で言えばサイクリックコード最近ではリードソロモン符号といったエラー検出／訂正符号を記憶しており、これを用いたエラー検出／訂正機能を備えている。このため 11 ビットバースト程度以内の小規模なデータエラーは訂正可能エラーとして自分で訂正し、それ以上のものは訂正不可能エラーとして確実に検出し、制御装置に報告する。これはその制御系についても同様のことが言える。即ち、現代のディスク装置では、データなり制御なりが狂っているにもかかわらず、あたかもそうでないように、自分が正常であるかのようにふるまう確率はきわめて少ない。このためディスク装置を制御する側から見ると、そのディスクが正常であるか否かが確実に把握できる。即ち、ディスク装置から異常報告があった場合ないしはタイムアウトエラーないしは制御装置自体が検出するインタフェースエラーのいずれでもなければ、このディスク装置は正常であると判断できる。このようにして、どのディスク装置が壊れているかがはっきりと特定できるため、RAID4 ではパリティという単純な冗長符号を付加するだけで、ディスク 1 台故障のエラー訂正が出来ることになる。

［2］分散 RAID4 への拡張

さて以上の RAID4 ディスクの故障復元機能を分散 RAID にも適用できないだろうか。これを検討したのが図 4.3 である。前章にて試作に成功した分散 RAID0 方式と、RAID4 ディスクアレイ方式から分散 RAID4 方式の導出を試みる。データ配置の観点では、図 4.3 から両者はぴったり同じデータ配置にすることが可能であり問題はない。制御法の問題については、ストライピングされたデータブロックを読み出す機能は、パリティサーバーがないものと考えれば、分散 RAID0 の時と同様であるため問題はない。1 台のサーバーが故障（ないしはそのディスクが故障）した場合の読み出しについては、その故障したサーバーから読み出すかわりに、今度はパリティサーバーを読みに行かねばならない。そして、図 4.3 の分散 RAID4 型ビデオサーバーにおいて、例えばサーバー1 が故障したとすれば、読み出した D0 と P0 を基に、

D1＝D0　xor　P0

の Exclusive OR 演算を実施して D1 のデータを復元し、これをクライアントに送らなければならない。これはビデオ放映中のリアルタイムデータ転送中のきわめてタイムクリティ

カルな処理である。このような高速処理は前章の分散 RAID0 で使った MS-DOS の TSR（Terminate and Stay Resident）機能を使う他ないと思われる。この場合、同様に TSR 機能を使っている分散 RAID0 で開発した“ストライピングドライバ”の中に RAID4 のこの機能を組み込めたら丁度都合がよいため、この方向で検討することとした。

次に故障サーバーの検出法については、試作機ではクライアント側で検出することとした。クライアントが１本のビデオプログラムを放映していて、あるサーバーに対するアクセスがエラーで返ると、このサーバーを故障サーバーとして記憶し、以後縮退運転モードに入る。以後この故障サーバーにはアクセスせず、かわりにパリティサーバーにアクセスする。もしこの放映中に故障サーバーの修理が完了してもそのサーバーにはアクセスしない。そのかわり、又別のビデオプログラムを放映する時点からこの修理完了したサーバーにアクセスするようし、正常運転モードに戻るようにした。

次にデータロードの問題については、前章で述べたのと同様タイムクリティカルな処理ではない。そのため TSR 機能を使う必要はなく、例えばクライアントの１つに RAID4 用データロードプログラムを作って載せることにより実現できる。 TSR 機能を使う必要がないため、データを複数のサーバーにストライピングしてロードしていくこと、パリティを計算してパリティディスクにロードしていくこと、といった機能は問題なく実現できる。

以上から分散 RAID4 型ビデオサーバーの実現性とその試作に対する見通しがついた。

４．２．２　試作の概要

以上の検討から分散 RAID4 型ビデオサーバーの試作は、3 章で行った分散 RAID0 型の試作に準じて行うこととした。今回はさらにデータに冗長性（パリティ）を持たせてパリティデスクを設けなければならない。パリティディスクは通常モードではアクセスされないが、故障が発生するとデータの復元のために使われる。このデータ復元機能は、分散 RAID0 で作成した“ストライピングドライバー”を改造して実現させる。これにより複数のサーバの内の 1 台が故障してもデータの喪失が起こらず処理を継続出来るようにする。

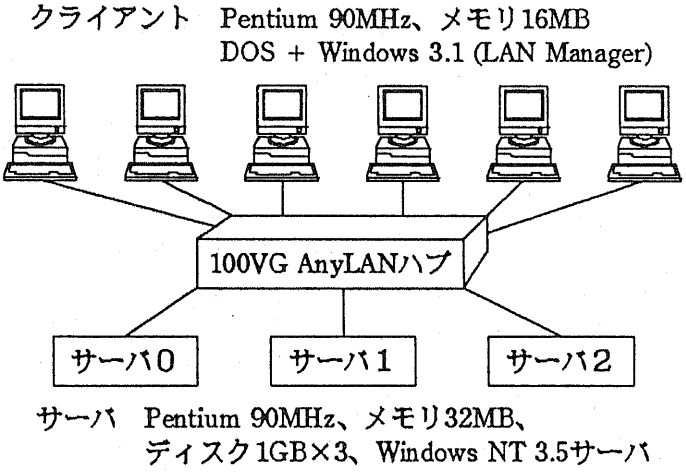


図 4.4　分散 RAID4 型ビデオサーバー試作機のシステム構成図

図 4.4 に試作システムの構成を示すが、これは 3 章で試作したものと同様である。LAN として 100Mbps イーサネット、各サーバにはディスクを 3 台接続しソフトウェアストライピング構成とする。又ソフトウェアによる実現法も分散 RAID0 型の時と同様、クライアント側に分散 RAID4 の機能を持たせ、サーバー側には何も手を加えない。

図 4.5 にソフトウェア構成を示すが、ここで変わっている点はサーバー3 がパリティサーバーになっており、それに伴ってストライピングが変更されていることである。ストライピングドライバは MS-DOS の TSR(Terminate and Stay Resident)機能を使って実現し、その中に分散 RAID4 型の機能が組み込まれる。

又前章で述べたのと同じ理由により、ここでもREAD機能だけを実装し、データの格納は別プログラムにより行うこととした。

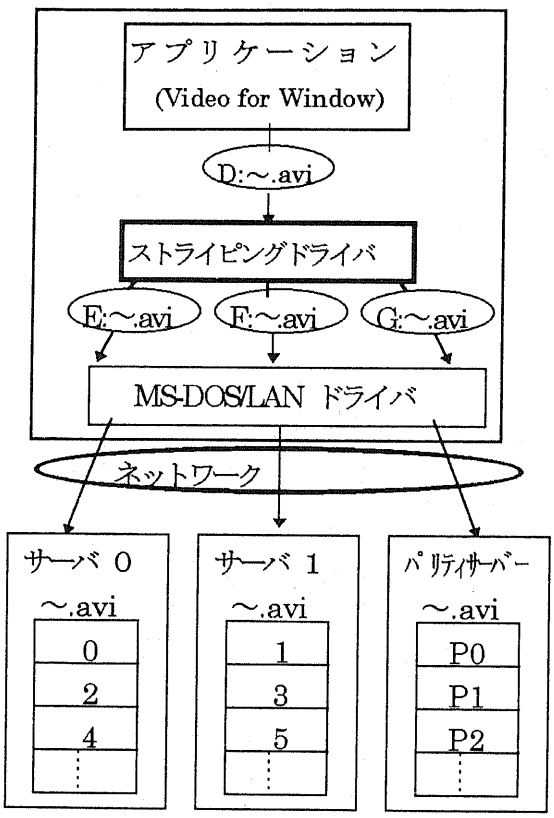


図 4.5　分散 RAID4 型のソフトウェア実現方式

なおRAID4型構成の特徴は、正常時には働かないパリティ専用サーバが出来ることである。そのためこのパリティ専用サーバにビデオ放映の前後処理他を管理するコントロールサーバを兼用させる等の利用法も考えられるが、試作機には実装しなかった。

４．２．３　分散ＲＡＩＤ４型ストライピング

分散 RAID4 型のストライピング方式として、分散 RAID0 と同様サーバ当たり複数台のディスク接続が可能な方式を考える。ストライピング手法としてごく自然に発想されるものは図 4.6、図 4.7 に示すものであるが、図 4.8 のような方法も考えられる。

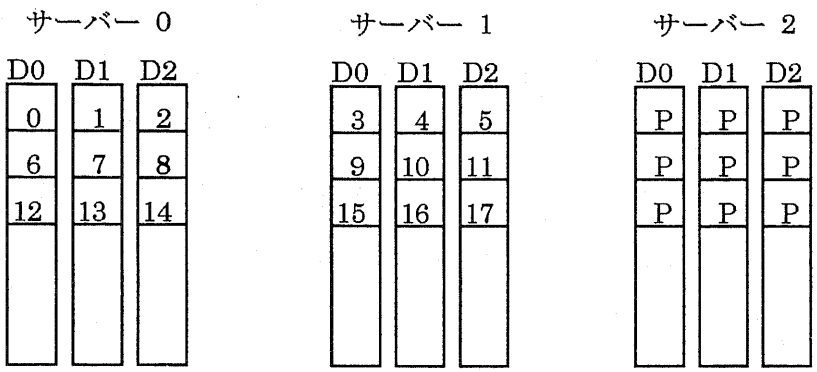


図 4.6　分散 RAID4 のストライピング方式（その１）

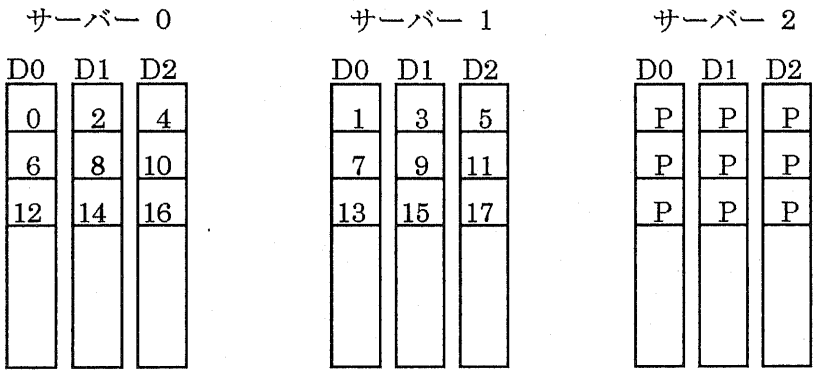


図 4.7　分散 RAID4 のストライピング方式（その 2）

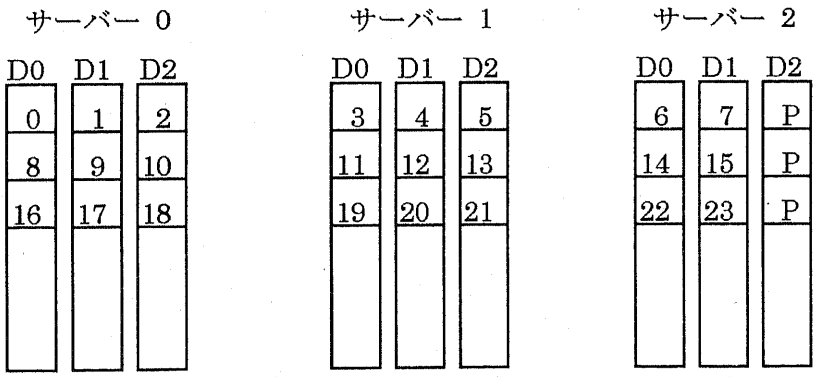


図 4.8　分散 RAID4 のストライピング方式（その 3）

図 4.6 はストライピングされるブロックが順次サーバを移っていく方法であり、図 4.7 は順次ディスクを移っていく方法である。一方図 4.8 は横一列のストライピングに対してパリティブロックを 1 つだけ置く方法であるが、この場合はディスクが 1 台故障した場合にのみデータ復元が可能となる。ここでは広くサーバー故障まで取り扱うこととし、図 4.8 の方法は除外した。さらに、後述するストライピングバッファ容量が少なくて済むこと、プログラムがサーバー当たりのディスク台数を意識しなくてよい等の長所から図 4.7 の方法を採用することとした。

４．２．４　ストライピングバッファ

図 4.9 と図 4.10 は試作した分散 RAID4 型ビデオサーバーの動作を示している。これらの図において、データがクライアント中に置かれたストライピングバッファを経由して流れるのが特徴である。ストライピングバッファはストライピングの横一列分のデータを貯えるバッファである。図 4.9 は正常時の動作を示すが、このストライピングバッファを経由することとパリティサーバーにアクセスしないことを除いて、分散 RAID0 型の時と同様に動作する。一方図 4.10 はサーバー0 が故障して切り離された状態（縮退運転）の動作を示しており、サーバー0 の代わりにパリティサーバーがアクセスされる。ストライピングバッファ上に読み込まれたサーバー1 のデータとパリティの間で Exclusive OR 演算が行われ、サーバー0 のデータが復元される。

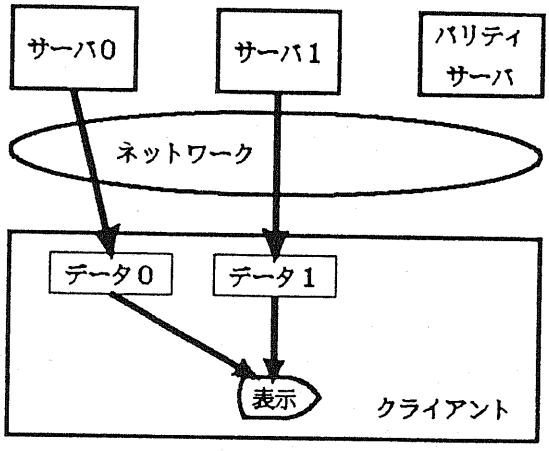


図 4.9　正常運転時のデータの流れ

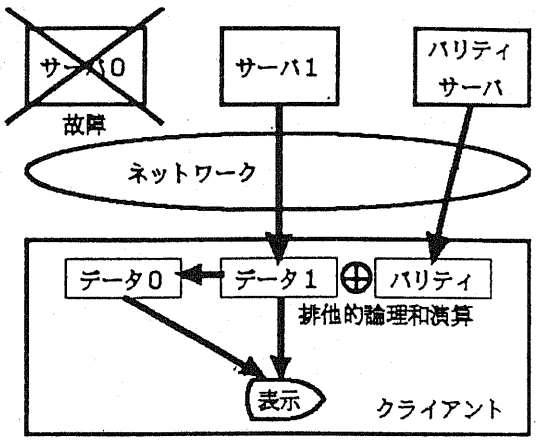
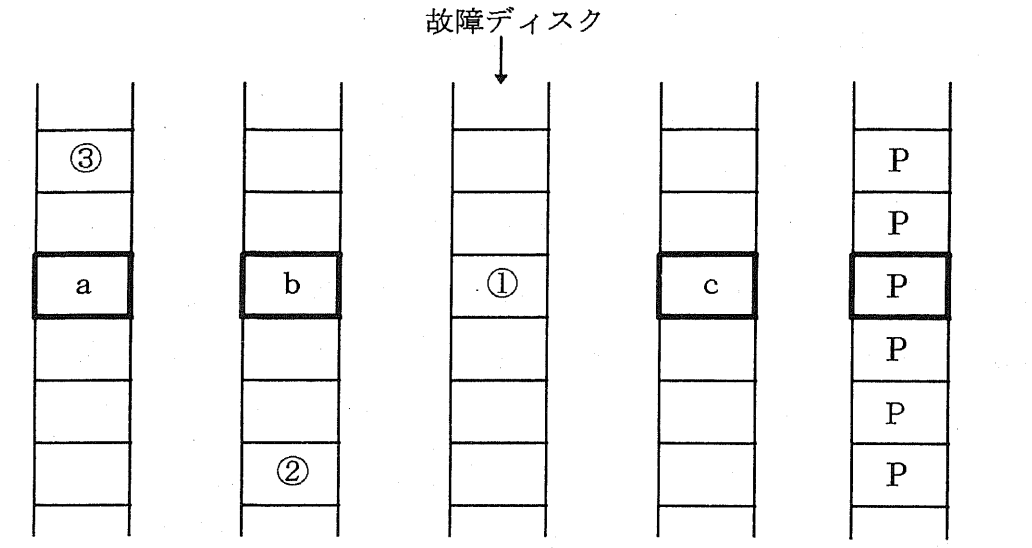


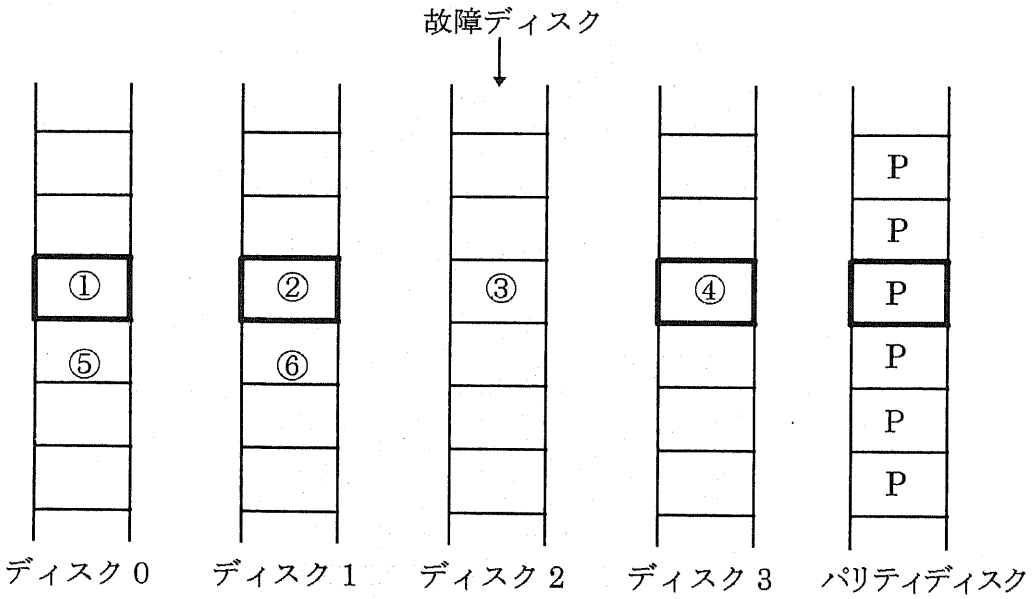
図 4.10　縮退運転時のデータの流れ



・データは①、②、③・・・の順にアクセスされる。

・①のデータを復元するために、読まれた a、b、c のデータは捨てられる。

図 4.11 ランダムアクセス時の RAID4 のデータ復元



・データは①、②、③・・・の順にアクセスされる。

・③のデータを復元するために、ストライピングバッファに貯えられた

通常に使われるデータ①、②、④が利用できる。

図 4.12 シーケンシャルアクセス時の RAID4 のデータ復元

上記の縮退運転時に、ビデオデータの特性を生かした高速化が図られている点に注目願いたい。図 4.11 に示すように一般にトランザクション処理等では、ファイル上のデータはデータブロック単位にランダムにアクセスされる。これに対しビデオデータは、図 4.12 に示すようにファイル上をシーケンシャルにアクセスされる。このため前者ではデータを復元するために読みとられた、同一パリティグループの他のデータブロックが再利用されることはほとんどなく、単に捨てられる（図 4.11）。これに対しビデオデータは連続的アクセスのため、上記パリティグループのデータブロックはパリティを除いて再利用されるという大きな利点がある（図 4.12）。図 4.10 においてデータ 1 は最初データ 0 の復元を行う目的で読み出されて、復元が行われデータ 0 が放映される。ここでサーバーから読み出されたデータ 1 はこの後もストライピングバッファ上に止まっており、データ 0 の次にデータ 1 が放映される番になると、今度はこれが放映データとして使われる。即ちここではサーバーから読み出さなくてもよいため 1 回得したことになる。即ち、ビデオデータではその順次アクセス性から、クライアント上に 1 パリティグループ分のストライピングバッファを設けることにより、本来 2 回の読み出しが必要のところを 1 回に削減出来る。図 4.9 と図 4.10 を比較すると、両者においてサーバから読み出してネットワーク上を転送する回数は、それぞれ 2 回ずつで同じである。即ち、本研究で対象としているサーバーとネットワークの系から見た性能（ビデオストリーム供給能力）は、正常運転時と縮退運転時で差がないことが分かる。

この他にクライアント側では故障復元のために Exclusive OR 演算処理が必要となるが、このための負荷増加は 10 数%と見積もられ、大きなものではない。3.2.1 で述べたようにこれは大きな問題ではなく、又本研究の対象外であるため、これ以上深入りしない。

注）ここで述べたストライピングバッファの事項は、米、英、独と日本に特許出願中であるが、このうち英国では UK Patent GB 2299424 B として 1997.3.5 に特許化された。

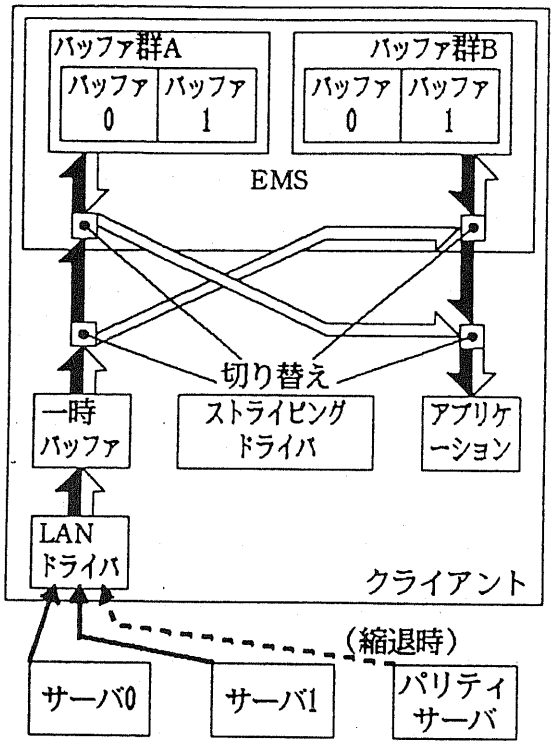


図 4.13　ストライピングバッファの実現法

ストライピングバッファは図 4.13 に示すようにクライアントの MS-DOS・EMS(Expanded Memory Specification)メモリ上に 2 つのバッファ群として構成される。サーバ数を S とすると、バッファ群は(S-1)個のバッファからなり、サーバから読み出された 1 ブロック(16KB)のデータが 1 バッファに読み込まれていく。MS-DOS の制約からこれは 1 バッファずつシリアルに行われ、バッファ群にデータが揃ったところでアプリケーションへの転送が可能となる。この間もう一つのバッファ群からはアプリケーションにデータが転送されダブルバッファリング処理が行われる。パリティサーバ以外のサーバが故障した縮退運転中は、故障サーバからのデータが入るべきバッファにパリティサーバからのパリティが読み込まれ、(S-1)個のバッファ間で Exclusive OR 演算を行って故障サーバのデータが復元され、結果が上記パリティが読み込まれたバッファに上書きされ、アプリケーションプログラムに渡される。

以上のようなストライピングバッファの働きにより、サーバ側の縮退時の性能を正常時と同一に保つことが出来る。

４．２．５　性能評価

以上に述べた分散 RAID4 型ビデオサーバを試作し性能評価を行った。性能評価環境は分散 RAID0 型の時と同様である。即ち図 4.4 のシステム構成で、ビデオデータは Video for Windows に付いてきた AVI 形式の、1.2Mbps の約 5 分のビデオデータを使用した。又同様に 1 台のクライアントではビデオデータをサーバー群から読み出して放映し、コマ落ち率の測定を行うが、他のクライアントでは疑似プログラムを走らせて、１つのクライアントからあたかも多数のクライアントがアクセスしているようにサーバー群に対して読み出しを行うようにし、クライアント数の不足を補った。コマ落ち率の測定は同じく Video for Windows に付いて来たテストプログラムを使用した。

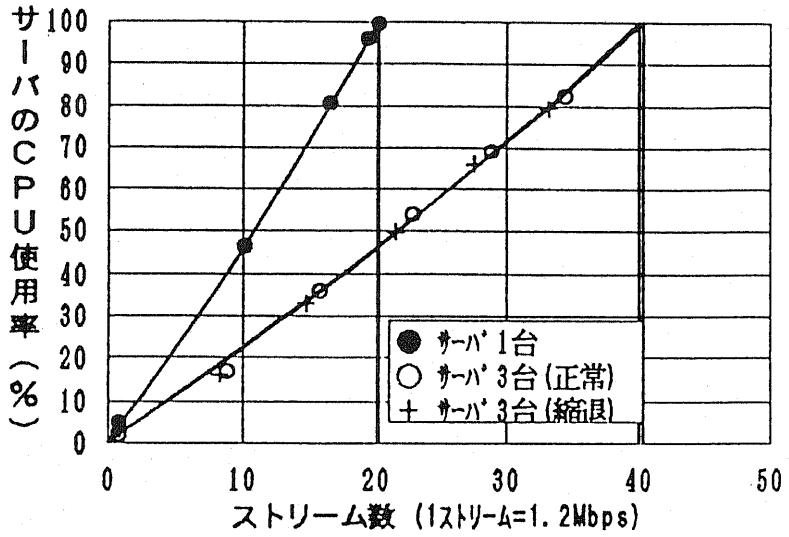


図 4.14　分散 RAID4 型ビデオサーバーの性能評価

図 4.14 がこの結果であり、正常時と縮退時の２つのケースにつき、ビデオストリーム数を増やしていった時のサーバ CPU の使用率が示されている。このグラフから正常時と縮退時の性能がほとんど変わらないことが達成されていることが判る。最大供給ビデオストリーム数（サーバ CPU=100%の点）はそれぞれ 40.4、40.0 と近く、これは RAID0 型 1 サーバの 20.2 に対して約 2 倍の性能となっている。以上、当初の目標である、縮退時の性能が落ちないこと、サーバ数に比例した性能向上の 2 点を達成していることを確認した。

表 4.1 は図 4.14 中の各測定点に対応した、放映クライアントでのコマ落ち率を示したも

のである。

表 4.1　コマ落ち率の測定

ｹｰｽ	ｽﾄﾘｰﾑ数	ｺﾏ落ち数 (5128 ﾌﾚｰﾑ中)	ｺﾏ落ち率
RAID4 正常 3 ｻｰﾊﾞ	1.0	0	0.00 %
	9.0	1	0.02 %
	15.9	1	0.02 %
	22.8	0	0.00 %
	28.9	0	0.00 %
	34.3	0	0.00 %
RAID4 縮退 3 ｻｰﾊﾞ	1.0	1	0.02 %
	8.4	0	0.00 %
	14.9	2	0.04 %
	21.5	2	0.04 %
	27.6	0	0.00 %
	33.1	0	0.00 %

この表から上記測定時のコマ落ち率は 0%ないし、多くとも 0.04%と少なく、良好な映像が転送されていることが判る。又、少なくとも 1 台のサーバが故障して切り離された縮退運転中に、33 ストリームのビデオデータを供給し、コマ落ち率 0%で良好な映像であることが事実として確認出来たことも大きな成果である。この時クライアント側 CPU はデータ復元のために Exclusive OR 演算を実行するが、これは 10%強の CPU 負荷であり、クライアント CPU にとって大きな負担とはなっていない。

なお疑似故障の発生法は、故障させるサーバの共有ディレクトリサービスを停止させ、突然クライアント側からアクセス出来ないようにすることにより行った。

この操作で故障発生させた場合には、放映クライアントの映像には全く影響を与えずに、故障サーバの切り離しと縮退モードへの切替えが行われることを確認した。

4. 2. 6　まとめ

これらの結果から、試作した分散 RAID4 方式ビデオサーバーでは、サーバー数に比例した

スケーラブルな性能向上がほぼ実現出来ることが確認された。これは理論通りと言えばそれまでだが、サーバーのH/W（CPU、ディスク）、S/W（OS，通信ドライバー他）の処理性能、LAN上の振る舞い等の種々の要因がからむ中で、実際に検証されたことの意義は大きい。試験は数10ビデオストリームという実際に近い環境で行われれ、映像品質も目視やコマ落ち率測定プログラムにより確認した。又1サーバーの故障挿入試験も行い、その時にクライアント側の映像の乱れ無しに、縮退運転モードに移行して、正常に運転が継続されることも確認した。

以上の結果から、分散RAID4方式ビデオサーバーの実現性が確認され、実用化に向けて大きく前進したと考える。

４．３　分散ＲＡＩＤ５型ビデオサーバーの提案とシミュレーション評価

４．３．１　正常動作時の性能向上

分散 RAID4 型ビデオサーバーでは正常動作時に S 台のサーバの内パリティ以外の(S-1)台がオペレーションを実行するが、RAID5 型ではパリティが固定されておらず S 台のサーバーが動作出来るため、ビデオ供給性能が向上することを発見した。この場合の性能向上率は S/(S-1)と予測出来る。但し、RAID4 型では故障発生時にも正常時と比較して性能低下が無かったが、RAID5 型では故障発生時には(S-1)台のサーバのみで動くことになり、正常時に比べ(S-1)/S だけ性能が低下する。

同一のハードウェア量でありながら正常時の性能が RAID4 よりも向上する点は RAID5 型の大きな魅力である。

４．３．２　分散ＲＡＩＤ５型ストライピング

分散 RAID5 型のアレイ構成法として、ごく自然に発想される方法は図 4.15、図 4.16 に示すものである。

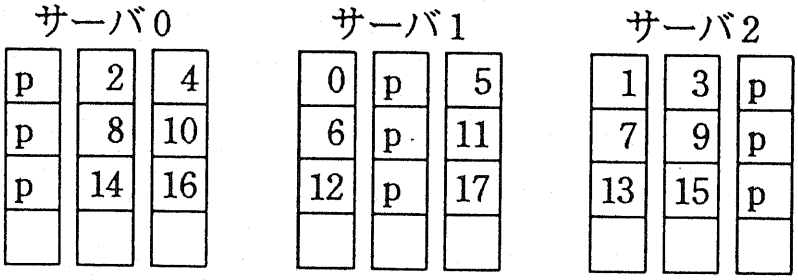


図 4.15　分散 RAID5 型のアレイ構成法（その 1）

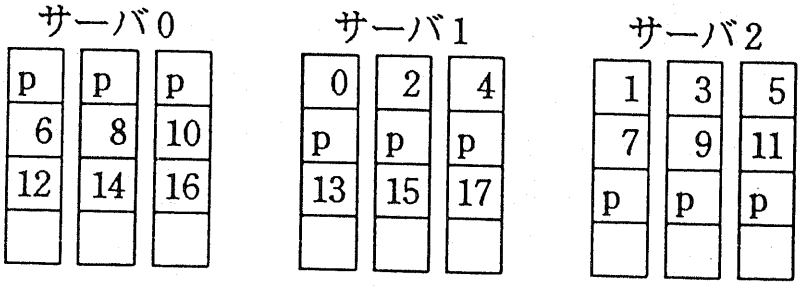


図 4.16　分散 RAID5 型のアレイ構成法（その 2）

図 4.15 の方法は、パリティグループ毎にパリティの位置を 1 つずつずらしていく方法である。図 4.16 の方法は、サーバー当たりのディスク台数を D とすると、D 回連続して同一サーバーにパリティを持たせる方法である。サーバー数を S とすると、図 4.15 の方法では D＝nS のケース（整数倍）で、パリティだけを持つディスクが出来てしまうという欠点がある。図 4.15 は n＝1 の場合のこの例であり、各サーバ内で 1 つのディスクにパリティが集中するため、正常動作時にこれらのディスクは遊んでしまいディスクネックの状況において性能低下を来す。又逆に nD＝S のケース（整数倍）でもパリティが特定のディスクに片寄るという欠点がある。一方図 4.16 の方法はプログラムがディスク台数 D の値を意識しないといけない欠点がある。プログラムの複雑さよりも実利（性能）の方をとって図 4.15 を採用することとした。

４．３．３　シミュレーションによる性能評価

以上のアレイ構成に基づき、分散 RAID5 型ビデオサーバーをシミュレーションにより性能評価した。又、比較のため分散 RAID4 についても同様の方法でシミュレーションを行った。このシミュレータは C 言語で 1332 ステップで構成され、UNIX 上で動作する。シミュレーションはディスクについては、シーク時間、回転待ち（共に確率はランダム）ディスクリード時間等詳細に行われるが、CPU 処理時間、ネットワーク転送時間については大まかに近似的かつ均一に行っている。そのためディスクネック状況のシミュレーションに適したものとなっている。本シミュレータの特徴はコマ落ち率をシミュレーションしていることである。クライアントが必要時点よりも遅れてデータブロックを受け取った場合、最低次の 1 ブロックはスキップして（それ以上遅れている場合はその分、数ブロックスキップして）次のブロックに対してリクエストを出す。スキップした画面数をこの場合のコマ落ち画面数として計算する。このようにしてコマ落ちのシミュレーションが行われる。シミュレーションに使用したパラメータは次の通り。

サーバー数＝3、サーバー当たりのディスク数＝3、ディスクの回転待ち＝0～11msec、データブロックサイズ＝64KB、ビデオデータレイト＝1.2Mbps、ビデオデータサイズ＝1000 ブロック、ビデオプログラム本数＝15

シミュレータの構造は、大きくビデオサーバーと複数のクライアント、そしてそれらを

つなぐ通信路からなっている。クライアントはそれぞれの放映スケジュールに照らして、事前にサーバーの一つに次のデータ要求を出す。これらは通信路伝播時間（一定）の後に該サーバーの入力キューに入る。そのサーバーの CPU はこれらを順番に処理する（処理時間一定）。各ディスクにも入力キューが存在する。前記の“CPU の処理”とはこのデータ要求を所定のディスクの入力キューに入れることである。各ディスクは、その入力キューからデータ要求を取り出し、ディスク読み出しをシミュレートする。この時は前述のようにシーク時間、回転待ち時間はランダムに設定される。ディスクの読み出しが完了すると、通信路を介して（伝播時間＝一定）、データがクライアントに到着する。クライアントはこのデータが放映に間に合ったか否かチェックし、コマ落ち率を計算する。

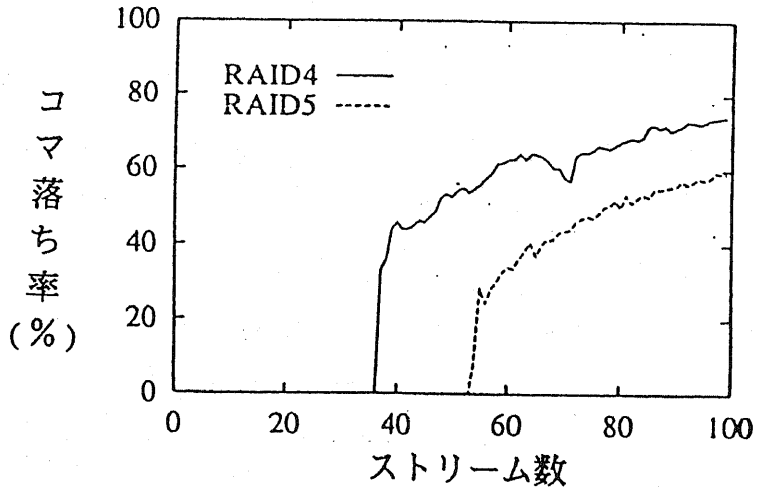


図 4.17　分散 RAID5 ビデオサーバーのシミュレーション結果（1）

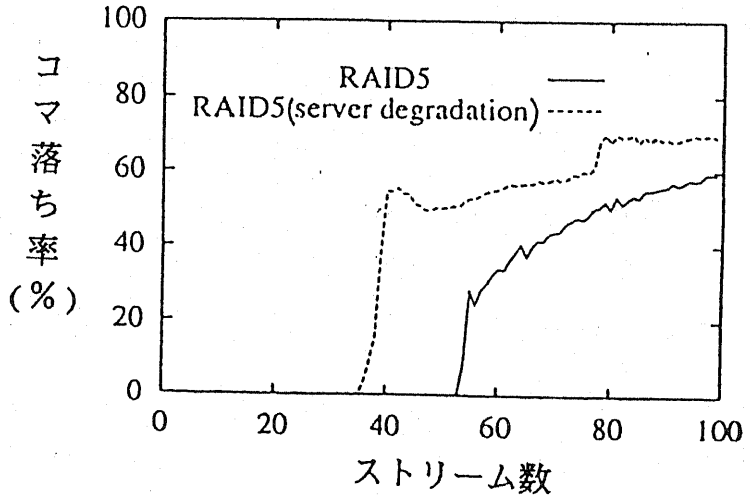


図 4.18　分散 RAID5 ビデオサーバーのシミュレーション結果（2）

図 4.17 と図 4.18 にシミュレーション結果を示す。これらのグラフで横軸はビデオストリーム供給数、縦軸はコマ落ち率を示す。図 4.17 は分散 RAID4 型と分散 RAID5 型の通常動作時を比較したグラフである。ビデオサーバーのビデオ供給能力はコマ落ち率が 0 から正の値に変わる点と見ることにすると、図 4.17 において分散 RAID5 型のビデオ供給能力は 53 ストリーム、分散 RAID4 型のそれは 35 ストリームとなる。この比 53/35=1.51 は、この時のそれぞれの動作サーバー数の比 3/2=1.5 とほぼ一致することが確認出来る。一方、図 4.18 は RAID5 型の、通常動作時と縮退時を比較したものである。この場合の両者のビデオ供給能力は通常動作時が 53 ストリーム、縮退時が 35 ストリームと読みとられる。この比 53/35=1.51 も同様に動作サーバー数の比 3/2=1.5 にほぼ一致することが確認出来る。以上の結果から、通常動作時には分散 RAID5 の方が分散 RAID4 よりビデオ供給能力が高いこと、縮退時には両者共同じになること、またこれらの値はその時の動作サーバー数に比例すること（計算と良く一致すること）、がシミュレーションにより確認出来た。

４．４　分散ＲＡＩＤ５におけるディスクレベルの縮退方式の提案

従来の分散 RAID 方式［Cabrera-91］［Long-94］ではサーバ単位の縮退のみが提案されていた。ここでは分散 RAID5 型に限って、ディスク単位の縮退も可能とすることによりさらに可用性を増す方式を提案する。可用性（Availability）を増すとはここでは縮退運転時の性能低下を減らすことを意味する。4.5 にて述べたように分散 RAID5 型ではサーバー数を N とすると、縮退運転時には正常運転時に比べて（N－1）／N 倍に性能が低下する。この性能低下量を減らそうというのが本節の趣旨である。

４．４．１　ディスク単位の縮退

上記のように故障発生時にサーバーごと切り離すと、分散 RAID5 では性能が（N－1）／N 倍に落ちることは避けられない。ここでもしその故障が、そのサーバー中の 1 台のディスクであるならば、サーバーは切り離さず、そのディスクだけを切り離そうというものである。これにより性能低下は（N－1）／N よりも低く押さえられることは容易に想像できる。但しこの手法が適用できるのは以下の場合に限られる。

・分散 RAID5 型の場合

∵分散 RAID4 型では、正常運転時→縮退運転時で性能低下がないため意味がない。

・1 つのサーバーにディスクが複数台ついている場合。

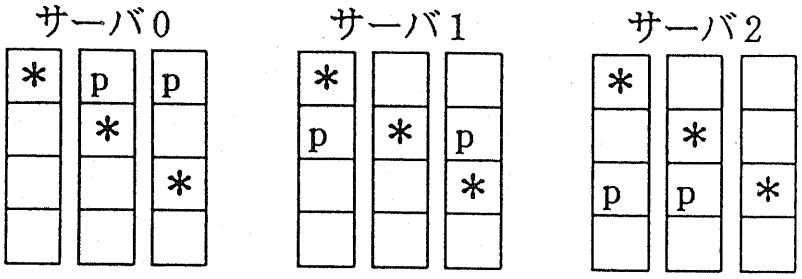


図 4.19　RAID ディスクアレーを使う方法

ディスク単位の縮退を実現するには、最も簡単な方法としては各サーバのディスクを、単に市販の RAID ディスクアレーに置き換える方法がある。この方法によれば、各ディス

クアレーの中で 1 台のディスクが故障した場合には、その中で故障した 1 台のディスクが自動的に切り離されて縮退運転に入ってくれる。サーバーから見ると何も知らない間にこれ（ディスクレベルの縮退）が行われるので大変楽である。但しこの方法には次のような欠点がある。図 4.19 はこれを示したものであり、ディスク 3 台構成の RAID5 ディスクアレーが各サーバーに 1 個ずつ接続された例である。各ディスク中にストライピングされるデータブロックが桝目で示されている。その中で“＊”で示してあるのが RAID ディスクアレーで持っているパリティブロックである。上記の自動的なディスク縮退はこれを基に行われる。これに対し“p”で示されているのが、分散 RAID5 型の持っているパリティーである。これから分かるようにこの方法ではパリティブロックが多くなるという欠点がある。図 4.19 の例では 27 ブロック中 15 ブロック＝56%がパリティとなってしまう。これではディスク上のデータ効率が大変悪いと言わざるを得ない。

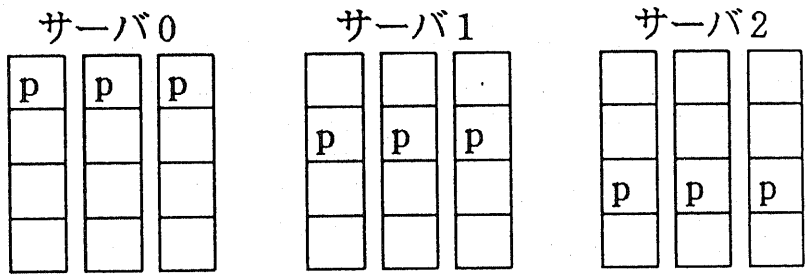


図 4.20　分散 RAID5 の中で実現する方法

図 4.20 はここで提案する方式のパリティの配置法を示しているが、これは前述の分散 RAID5 の配置法に他ならない。この場合には 27 ブロック中 9 ブロック＝33%がパリティとなり、上記に比べて少なくて済む。一般に、要素サーバ数を N、要素サーバ当たりのディスク数を D とすると、当提案の方式の方が D/(D+N−1)の比率でパリティが少なくて済む。

次にサーバ縮退に対するディスク縮退の利点を考えてみる。1 サーバが縮退した状況では D(N−1)台のディスクが稼働するが、1 ディスクが縮退した状況では(DN−1)台のディスクが稼働出来るため、ディスクネックの状況では(DN−1)／D(N−1)倍の性能向上が期待

できる。ディスクネックの場合に限らずサーバーネック（例えばサーバーのCPUネック）の場合も同様である。サーバー当たりディスクが3台ついているとすれば、故障したディスクを抱えるサーバーは、ディスク2台分については通常通り動作する。即ち通常の2／3の働きはすることになる。それ故サーバーネックの状況においても同様に(DN－1)／D(N－1)倍の性能向上が期待できる。

以上のように、ディスク故障が発生した時にサーバー全体を切り離すのでなく、そのディスクだけを切り離すことで上記の性能上の利点が得られる。

4．4．2　正常運転時と縮退運転時の性能のまとめ

本項では、それぞれ分散RAID4、分散RAID5、分散RAID5でディスク規模の縮退がある場合について、正常運転時と縮退運転時の性能について整理を行う。これをまとめたのが表4.2である（表中Nはサーバー数、Dはサーバー当たりのディスク台数を表す）。

表4.2　分散RAIDにおける稼動サーバー数／ディスク数

	Normal mode		Disk deg. mode		Server deg. mode	
	servers	disks	servers	disks	servers	disks
Distributed RAID4	N−1	(N−1)*D	−	−	N−1	(N−1)*D
Distributed RAID5	N	N*D	N	N*D−1	N−1	(N−1)*D

この表は次のように読むことが出来る。

分散RAID4の行で、正常運転時と縮退運転時の性能を比較すると、稼動サーバー数はN－1→N－1、全体の稼動ディスク数は（N－1）・D→（N－1）・Dと変わらないため、分散RAID4型では正常運転時も縮退運転時も性能は変わらないことが分かる。

次に表中、正常運転時の列を見て、分散RAID4と分散RAID5を比較すると、稼動サーバー数はN－1→N、全体の稼動ディスク数は（N－1）・D→N･Dとなり、分散RAID5の方が分散RAID4よりも、正常運転時の性能が良いことが分かる。

次に表中、分散RAID5の行で正常運転時とサーバー縮退時を比較すると、稼動サーバー

数はN→N－1に、全体の稼動ディスク数はN·D→（N－1）·Dと減るため、分散RAID5では正常運転時に比べて、縮退運転時に性能が低下することが分かる。

最後に、分散RAID5の行に注目して、サーバー縮退時とディスク縮退時を比較すると、稼動サーバー数はN－1→Nに、全体の稼動ディスク数は（N－1）·D→N·D－1に増えるため、サーバー縮退時に比べてディスク縮退時の方が性能が良いことが分かる。

４．４．３　シミュレーションによる評価

上記のように分散RAID5でディスク縮退機能を設けると、サーバー全体を縮退する場合に比べ性能低下率を減らせるはずである。この事実を確認するためにシミュレーションによる評価を実施することとした［清水-96］。シミュレータは4.3で作成したものにディスクレベル縮退の機能を追加する形で作成した。分散RAID5のアレイ構成は4.3の場合と全く同様とした。なおこのシミュレータはディスクボトルネック環境をシミュレーションするものである。CPUボトルネック環境もシミュレーションできるのが望ましいが、市販ソフトウェアの中味が分からないこともあり、非常に難しい。極限を追求した場合にはCPUボトルネックが問題であるが、通常のシステム構成ではディスク台数が不足して、これがボトルネックになることは良くあることである。上記のシミュレーションの評価値はこのディスクボトルネック構成における評価値である。

さてシミュレーションを行ったところ意外な結果が得られた。それはディスク縮退時とサーバー縮退時で性能が変わらないという結果が出たことである。

４．４．４　ストライピングの問題点

初めはシミュレータの不備ではないかと種々調査したが問題は見当たらなかった。さらに検討を進めた結果以下のような原因が判明した。

図4.21、図4.22、図4.23はこれを説明したものである。図4.21は、4.3.2の図4.16で示した分散RAID5のストライピング法にてストライピングした、分散RAID5のアレイ構成を示した図である。図4.21～図4.24はサーバー数が3台で、各サーバーにはそれぞれD0～D3の4台のディスクが付く構成である。ここでは1つのビデオプログラムがブロック0、1、2、3、・・・の順にストライピングして配置されている。図4.21は正常時のアクセス

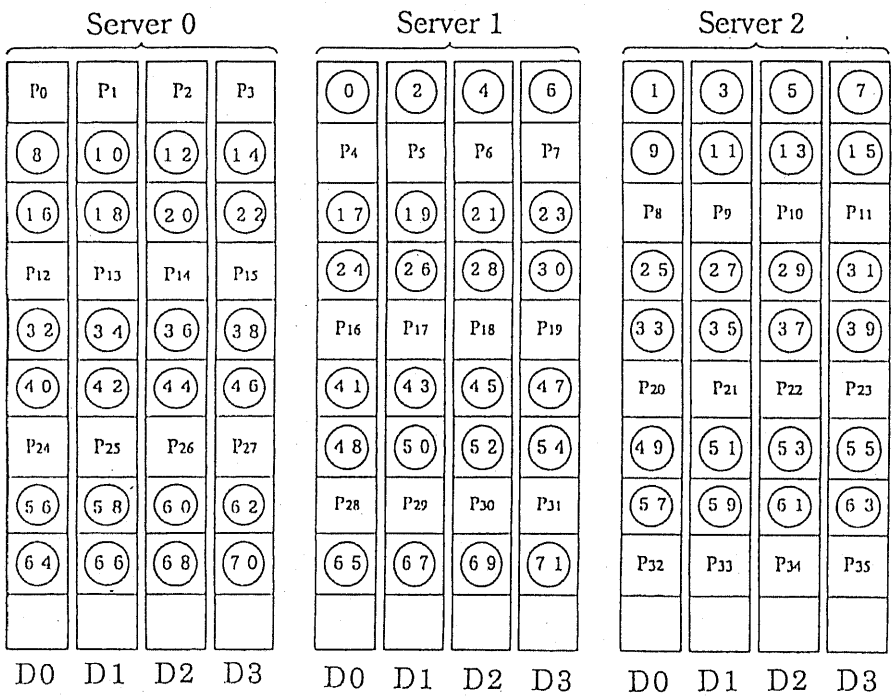


図 4.21　正常時のディスクアクセス

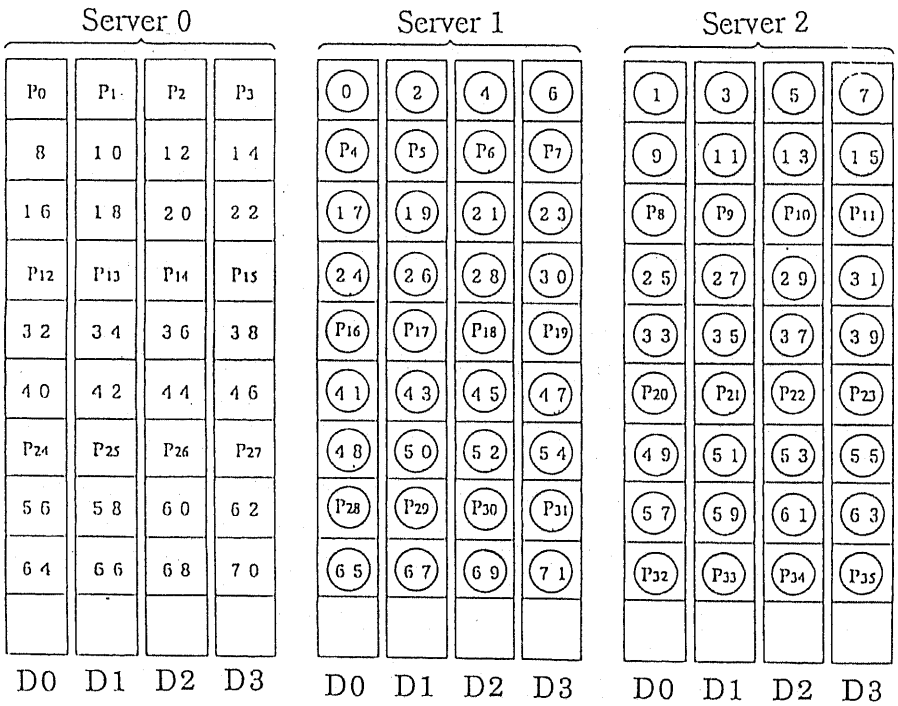


図 4.22　サーバー縮退時のディスクアクセス

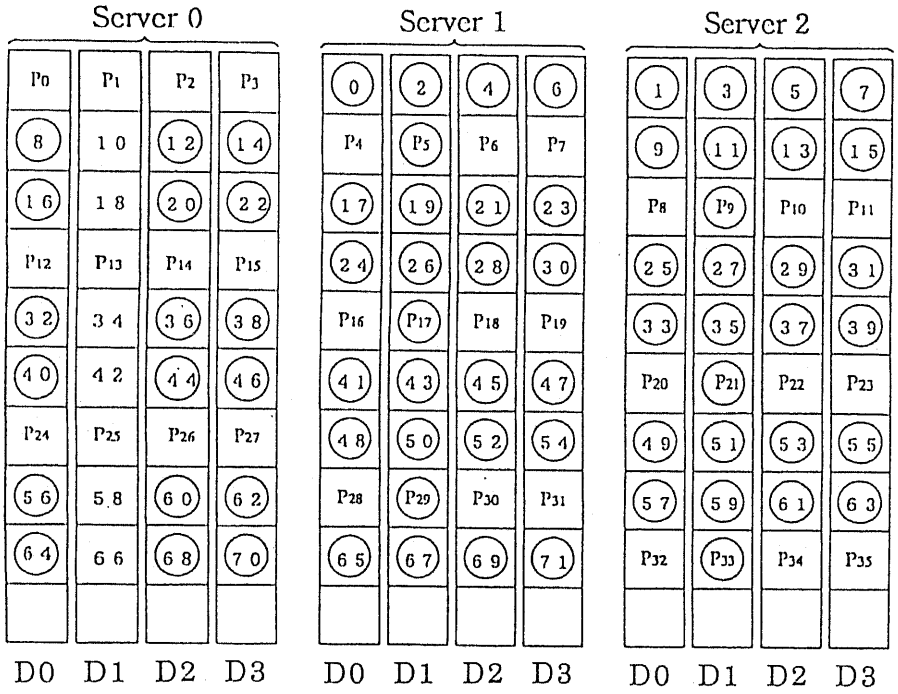


図 4.23　ディスク縮退時のディスクアクセス

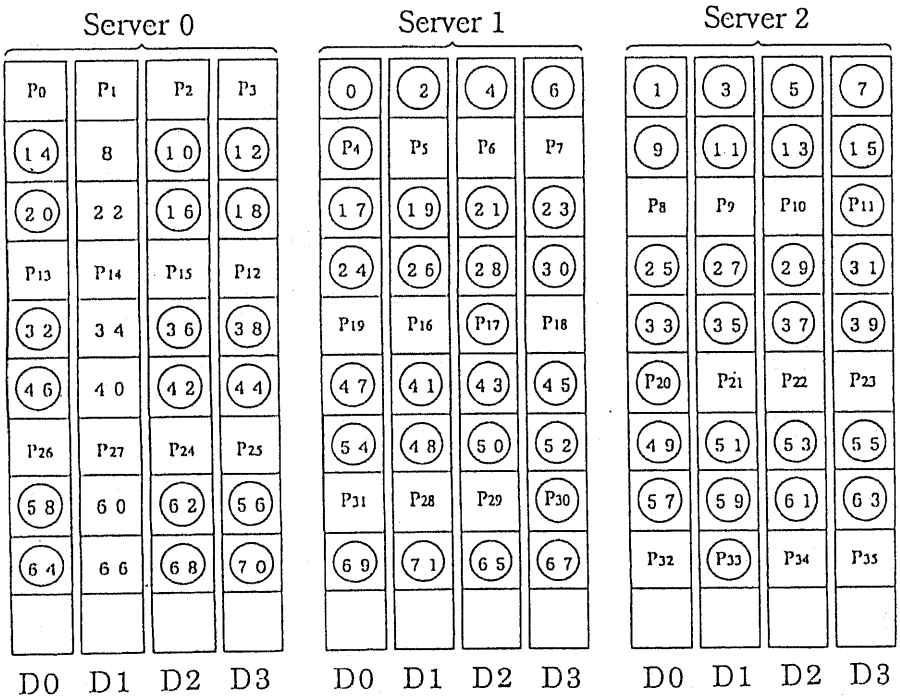


図 4.24　改良されたディスク縮退時のディスクアクセス

を示しており、丸印で囲まれたブロックが順にアクセスされることを示している。一方図4.22はサーバー縮退時のアクセスを示している。サーバー0が故障した場合で、サーバー0のディスクが全くアクセスされないかわりに、サーバー1とサーバー2に対するアクセスが増えていることが分かる。

さて、図4.23がいよいよ問題のディスク縮退時のアクセスを示したものである。ここではサーバー0は一定動いており、サーバーレベルでは一定負荷分散が行われていることが分かる。即ち、図4.22のサーバー縮退の場合と比べて、他のサーバーの負荷が軽減されているため、CPUボトルネックの状況に対しては有効なことが分かる。ところがディスクボトルネックの状況では、サーバー1とサーバー2のディスクD1に負荷が集中していることが分かる。即ちサーバー0の故障したディスクD1のデータを復元するために、これら2つのディスクだけが使われている。他のディスクには余裕が出来てもこれら2つのディスクの状況は、図4.22のサーバー縮退の時と変わっていない。このためディスクボトルネックの環境では、これら2つのディスクの状況によって押さえられるため、サーバー縮退からディスク縮退になっても、性能が変わらないことになる。これをまとめると次のようになる。

サーバー0～サーバー2をS0～S2、例えばサーバーS0のディスクD0をS0-D0と表せば、ストライピングされるグループは次の4つになる。（S0-D0，S1-D0，S2-D0），（S0-D1，S1-D1，S2-D1），（S0-D2，S1-D2，S2-D2），（S0-D3，S1-D3，S2-D3）。これから分かるように、各サーバーのD0ディスク群、各サーバーのD1ディスク群、・・が組み合わされている。故障の復元はストライピンググループの中で行われるため、図4.23に示されるようにサーバー0のディスクD1のデータの復元は、他のサーバーのそれぞれのディスクD1のデータを使って行われることになる。このように、ある1つのディスクのデータを復元するために、1部の特定のディスク群だけが使われる点に問題がある。

以上のように、4.3.2（図4.16）で用いた分散RAID5のストライピング法では、ディスクレベルの縮退を行った時に、ディスクボトルネック環境では、その効果が発揮されないことがわかった。

４．４．５　ストライピングの改良

上記のストライピング法の問題は、ストライピンググループを構成するディスクが、例えば（S0-D0，S1-D0，S2-D0）というように、毎回常に固定されることにある。この解決のためには、ストライピングの各行毎に、前とは違ったディスクの組み合わせになるようなストライピングにすればよい。このような方法としては例えば図 4.25 に示すようなマッピングテーブルを用いて、ディスクの組み合わせをずらしていく方法などが考えられる。

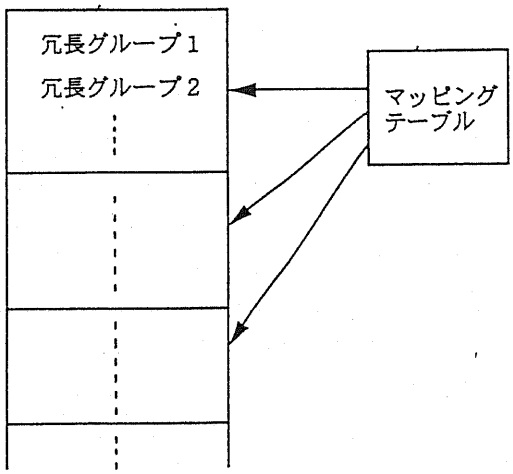


図 4.25　マッピングテーブルを用いる方法

ここでは、“ローテーション法”と呼ぶ、マップは使わず計算によって求められるストライピング手法を考案した［清水-96］ので、それについて説明する。

以下に図 4.21 で用いたアレイ構成、即ちサーバー数が 3、サーバー内のディスク数が 4 の場合を想定して説明する。図 4.26 は 1 つのサーバーの 4 台のディスクのローテーションを示している。0、2、4、6 はデータブロックの番号を意味し、図 4.21 のサーバー1 の 1 行目のブロック配置で代表させたものである。図 4.26 でローテーション回数が 0 の行は、データブロック 0 がディスク D0 に、データブロック 2 がディスク D1 に、データブロック 4 がディスク D2 に、データブロック 6 がディスク D3 に、割り当てられることを意味する。これは図 4.21 のサーバー1 の 1 行目のブロック配置と同様の配置である。これに対しローテーション回数が 1 のところでは、この配置を 1 つ右にサイクリックシフトした配置にな

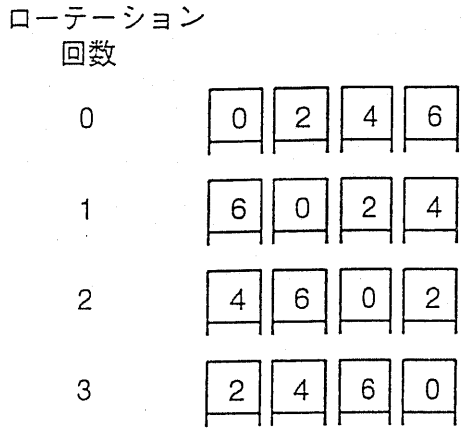


図 4.26　ローテーション番号の定義

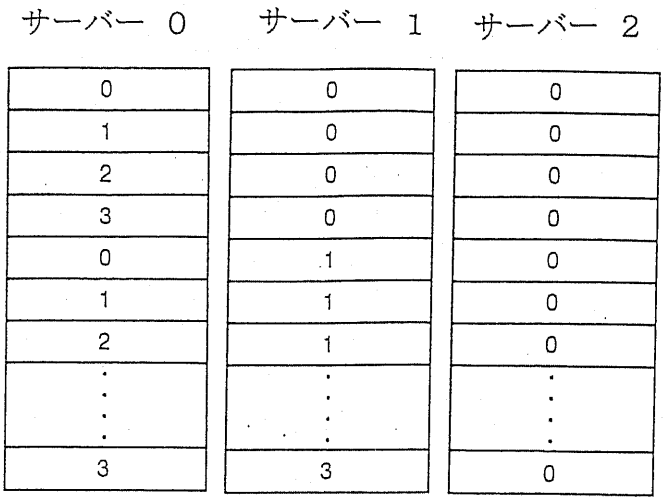


図 4.27　ローテーション番号の割り当て法

り、6、0、2、4 となる。同様にしてローテーション回数が 3 まで定義される。次に図 4.27 は各サーバーの各行の配置法を示している。この中に書かれている数字はローテーション回数を示している。図 4.27 を眺めると、右が上位桁の 3 桁の 4 進数が上から順番に並んでいることがわかる。以上が“ローテーション法”によるブロック配置法である。この方法によりストライピングしたのが図 4.24 である。図 4.27 で例えばサーバー0 の 3 行目は“2”となっており、即ちローテーション回数が 2 である。図 4.21 でサーバー0 の 3 行目は 16、18、20、22 のブロックが並んでいる。これにローテーションを 2 回行うと 20、22、16、18 となるが、図 4.24 のサーバー0 の 3 行目ではまさにそのような配置になっている。

以上の方法により、ストライピンググループを構成するディスクのグループが固定されなくなる。このストライピング法を適用した図 4.24 において、丸印を付けたブロックは、サーバー0 のディスク D1 が故障した場合にアクセスされるブロックを示している。図 4.23 と比較すると、サーバー1 とサーバー2 のディスク D1 に対する負荷の集中が解消されていることが分かる。即ち、これによってディスクボトルネック環境においても、ディスクレベルの縮退が効果を発揮出来るようになる。

なお、この方法にはまだ改良の余地が残されている。この方法により、サーバー0 のディスク D1 のデータを復元するために、サーバー1 とサーバー2 の全てのディスクが均等に使われるように改善された。しかしながら故障ディスクを持つサーバー0 のディスク D0、D2、D3 については、相変わらず復元に使われていないという問題が残っている。このためディスクボトルネック時における、ディスク縮退時の性能は、まだ理想値である（D·N－1）／D·N には達していない。この理想値を達成するためには、故障ディスクを持つサーバーのディスク群も故障復元に使われるようにローテーション法を改良する必要があり、今後の課題である。

注）以上の 4.4.3 と 4.4.5 の事項については、三菱電機（株）と静岡大学の共同出願で、日、米の 2 カ国に特許出願中である。

４．５　結言

前章にて試作した分散 RAID0 型ビデオサーバーは、サーバー数に比例して性能が向上する予測通りの好ましい結果を達成した。しかしサーバー数に反比例して信頼性が低下する欠点も併せ持っていた。本章ではこれを改善するための検討を行い、いくつかの改良案を提案し評価を行った。

4.2 節では分散 RAID4 型ビデオサーバーの提案を行い試作を行った。これはサーバー数を N とすれば、稼動サーバー数が常に（N－1）になるような冗長構成にして、サーバーの内の 1 台が故障しても、そのデータは復元され、そのサーバーを切り離して運転が続行されるようなビデオサーバーを実現する方式である。試作機を評価した結果、クライアントでビデオ放映中にサーバーの内の 1 台に故障を発生させると、このサーバーが自動的に切り離されて運転は正常に継続されること、なおかつこの切り離し時にクライアントの映像に乱れが起きないことが確認できた。又、分散 RAID 方式の本来の性質である、稼動サーバー数に比例した性能向上についても達成していることを確認した。

4.3 節では分散 RAID5 型ビデオサーバーの提案を行い、シミュレーション評価を実施した。分散 RAID4 型は正常動作時の稼動サーバー数が（N－1）台であるが、分散 RAID5 型ではこれが N 台になるため、性能が向上する利点がある。アレイ構成法等の実現方式を論じた後、シミュレーションによる評価を実施した。この結果予測通りの性能が実現出来ることが確認された。即ち正常動作時には、分散 RAID4 型に比べて N／（N－1）倍の性能を発揮すること、又縮退運転時には分散 RAID4 型と同等の性能であること、又サーバー数に比例した性能向上が図られることが、シミュレーション結果から明らかになった。

4.4 節では分散 RAID5 型において、ディスク単位で縮退する方式の提案を行った。もしディスクの故障であれば、そのサーバー全体を切り離すのでなく、故障したディスクだけを切り離すようにして、縮退運転時の性能低下を減らす方式である。考察の結果、この方式により、CPU ボトルネック環境においては所期の効果が発揮されることが確かめられた。一方ディスクボトルネック環境では所期の効果が発揮できないことが、シミュレーション結果から判明した。これは上記分散 RAID5 型で提案したストライピング法では、故障したディスクのデータの復元のために、一部の特定のディスクだけが使われるためと分かった。

この解決を図る方法の 1 つとしてローテーション法を提案しその効果を確認した。即ち故障ディスクのデータの復元が、1 部の特定のディスクでなく、他の多くのディスクを使って行われ、このためディスクボトルネック環境においても、ディスクレベルの縮退が効果を発揮することを確認した。

以上から、分散 RAID4 型、および分散 RAID5 型ビデオサーバーの実現性が明らかとなった。これにより分散 RAID 型ビデオサーバーの信頼性の問題が解決され、その実用化に向けて大いに前進した。

第５章　結論

１つのタスクを複数のプロセッサで処理するマルチプロセッサ技術は、汎用的な並列化手法が確立するまでに到っていず、応用毎に研究が行われているのが実状である。本研究ではビデオサーバー応用とデータベース応用を取り上げ、これらを最適に並列化して処理するマルチプロセッサシステムの構築を目指した。

第２章ではデータベースマシンについて論じた。ここでは主記憶とディスクをそれぞれ独立して持つ無共有型マルチプロセッサであるが、プロセッサ間を高速内部バスで接続し、共有メモリを介して高速通信することを特徴とする方式を提案した。本論文で提案した方式は、マスタープロセッサとスレーブプロセッサが共有メモリを介して高速通信するため、データベース応用には非常に適している。例えばスレーブプロセッサが“Selection”した多数のレコードをマスタープロセッサに送る場合に、共有メモリを使って高速に転送出来る。この他にも以下のような高速化の工夫を行った。

・RDB に特化したディスクアクセス方式を採用

・RDB に特化したディスクキャッシュ方式を採用

・ディスクキャッシュに直にアクセス（メモリコピー回数を減らす）

・水平分割方式のデータ配置法（3 カ国で特許化）

・プロセッサ毎に独立にディスクロード（ディスクバンド巾拡大）

・RDB の各処理をマルチプロセッサを利用して並列化

以上のような方式のデータベースマシンを試作した。その性能評価結果から、きわめて高性能を実現していることが明らかになった。例えば、製品化版の性能評価では他の同規模のデータベースシステムよりも約 40 倍高速である。又、このマシンの適用事例を紹介した中で、その高速性により、インデックスを使わないのみならず、無条件中間一致検索を使

うようにした事例で、①ユーザーにとって使いやすい、②データベースの設計、維持管理が省力化される、といった利点についても述べた。

以上から第２章で提案したデータベースマシンはきわめて有効な方式であると考える。

第３章と第４章ではビデオサーバーについて論じた。ここでは分散 RAID 方式ビデオサーバーと呼ぶ方式のマルチプロセッサ方式を提案した。これは主記憶とディスクを共有しない無共有方式のマルチプロセッサであるが、この無共有方式で問題となるプロセッサ間通信が、不要という特徴を有するものである。第３章では高性能化を目指した分散 RAID0 型ビデオサーバーを、第４章ではさらに高信頼化も実現する分散 RAID4 型ビデオサーバーを提案し試作した。これらの評価の結果、以下のような非常に好ましい特性を備えたビデオサーバーを実現出来ることが明らかになった。

・サーバー数にほぼ比例してビデオ配信数（性能）が向上する（スケーラビリティ）
・普及型標準品の安価なサーバーを複数使って実現出来る（価格性能比が大変良い）
・クライアント、サーバー、LAN（ないし通信網）は、全て市販品が使える。
　但し、クライアントには専用プログラムの追加搭載が必要。
・分散 RAID4 型では、複数サーバーの内の１台が故障しても、このデータは回復出来、
　そのまま運転を続行出来るという高信頼化機能を実現する。

このような数々の利点は分散 RAID 方式を提案した時点から予想出来たものである。しかし現実はなかなか理論通りには行かないものであり、試作して実際にそれを確認した意義は大きいと考える。評価においてはビデオ映像を目視チェックし、コマ落ち率を測定して確認し、又ビデオストリーム数が 43 本というような実システムに近いレベルまでも評価確認しており、十分説得力を持つと考える。

今後の課題としては、通信路の高性能化の問題が残されている。ビデオサーバーが高性能になっても、通信容量が不足しては結局性能は出ない。又、3.2 節にて行った実験結果から、映像品質に対する通信路性能の重要性も確認されている。試作システムでは通信路として 100Mbps の Ethernet-LAN を使用した。これに対し分散 RAID0 型では、サーバー数３台、ディスク全９台構成の規模で、総転送レート 51.4Mbps まで試験出来た。この後まだ伸びるとは思われるが、そろそろ限界に近いとも予想される。いずれにせよサーバー数を

倍の 6 台にすれば 100Mbps-Ethernet-LAN では不足することは明らかである。このための解決策として、そろそろ製品が出始めた Giga-bit- Ethernet に期待をかけたい。これは 100Mbps-Ethernet-LAN の 10 倍の通信容量を持つため上記問題の解決が期待出来る。

略語一覧

CCD: Charged Coupled Device

CD-ROM: Compact Disk-Read Only Memory

CPU: Central Processing Unit

C/S: Client/Server

DMA: Direct Memory Access

DRAM: Dynamic Random Access Memory

EISA: Extended Industrial Standard Architecture

HDM: High-speed Database Machine

H/W: Hardware

LAN: Local Area Network

LSI: Large Scale Integrated circuit

MPEG: Moving Picture image coding Experts Group

MPU: Micro-Processing Unit

MTBF: Mean Time Between Failure

OS: Operating System

PCI: Peripheral Component Interconnect

RAID: Redundant Arrays of Inexpensive Disks

RDB: Relational DataBase

SQL: Structured Query Language

SCSI: Small Computer Systems Interface

S/W: Software

TCP/IP: Transmission Control Protocol/Internet Protocol

TSR: Terminate and Stay Resident

VOD: Video On Demand

謝辞

本研究の過程において、直接懇切なるご指導とご鞭撻を頂いた静岡大学情報学部・水野忠則教授に、深く感謝申し上げる。また本論文をまとめる過程で種々適切なる御指導を頂いた静岡大学工学部・下平美文教授、同情報学部・渡辺尚助教授、同情報学部・中谷広正教授、同工学部・前田恭伸助教授に深く感謝申し上げる。

本研究は、三菱電機(株)情報技術総合研究所より社会人博士課程派遣学生として行ったものであり、本研究の契機を直接与えて頂き、かつ以前その元となる研究で御指導頂いた、当時の所長、現静岡大学・曽我正和教授に深く感謝申し上げる。その後見守っていただいた当時所長・現三菱電機(株)常務取締役開発本部長 野間口有博士、日々暖かく激励を頂いた、同情報技術総合研究所アーキテクチャ部長 風間成介氏、本研究にご理解とご援助を頂いている、同情報技術総合研究所・情報処理部門部門統括 岩瀬正氏、同所長室参事 数馬好和氏、第2章の研究の製品化で多大な御指導を受けた、名菱電子(株)取締役社長 服部訓明氏、同情報通信部長 伊藤正勝氏、同ソフトウェア研究室長 原紀久男氏、同情報システム課長 田中正明氏、当時三菱電機(株)半導体情報システムプロジェクトマネージャ 山本武夫氏、同北電グループマネージャ 丸田國貴氏、当時三菱電機(株)コンピュータ製作所副所長 坂和磨氏、同基盤技術部基盤技術課長 浜敬三氏、同主幹 馬場宏氏、同主事 松本利夫氏、同課長 大内博氏、三菱電機システムウェア(株)課長 板倉國司氏、同課長 花畑寿士氏、現三菱電機(株)開発本部技師長石田喬也氏、本研究に入る前に種々の助言、御指導を頂いた、現同本社システムプロダクト営業部長 富沢研三氏、現同情報通信システム開発センタ分散情報システム開発部長 上田尚純氏に深く感謝申し上げる。

研究の過程で同研究室にて、種々御指導頂いたインテック(株)テクニカルサービスセンター所長 田窪昭夫氏、三菱電機(株)情報システム製作所主管技師長 宮西洋太郎博士、同名古屋製作主管技師長 中野宣政博士、シミュレーションで本研究に協力を得た当時修士課程の清水洋氏に深く感謝申し上げる。

本研究の遂行においては、三菱電機(株)情報技術総合研究所主事 峯村治実氏に多大のご尽力を頂いた。ここに深く感謝の意を表する。

最後に、在宅の研究作業に協力を強いた家族に感謝する。

参考文献

[Babb-79] Babb, E., "Implementing a Relational Database by Means of Specialized Hardware," ACM Transactions on Database Systems, Vol.4, No.1, pp.1-29, March 1979.

[Baker-91] Baker, M.G., Hartman, J.H., Kupfer, M.D., Shirriff, K.W., and Ousterhour, J.K., "Measurements of a distributed file system,"in Proc. of the 13th ACM Symposium on Operating Systems Principles, pp.198-212, Association for Computing Machinery SIGOPS, October 1991.

[Baru-95] Baru, C.K., Fecteau, G., Goyal, A., Hsiao, H., Jhingran, A., Padmanabhan, S., Copeland, G.P., Wilson,W.G., "DB2 Parallel Edition," IBM SYSTEMS JOURNAL, VOL 34, NO 2, 1995.

[Bitton-83] Bitton, D., DeWitt, D.J., and Turbyfull, C., "Benchmarking Database Systems- A Systematic Approach," Proc. of the 18th VLDB Conference,1983.

[Cabrera-91] Cabrera, L.F. and Long, D.D.E., "Swift :Using Distributed Disk Striping to Provide High I/O Data Rates," Computing Systems, vol.4,Fall,1991.

[Codd-70] Codd, E.F., "A Relational Model of Data for Large Shared Data Banks," Comm. ACM, Vol.13, No.6, pp.377-387, June 1970.

[Cohen-95] Cohen, A., Burkhard, W.A. and Rangan, P.V., "Pipelined Disk Array for Digital Movie Retrieval," In Proc. of the IEEE International Conf. on Multimedia Computing and Systems, pp.312-317, May 1995.

[DeWitt-79] DeWitt, D.J., "DIRECT - A Multiprocessor Organization for Supporting

Relational Database Management Systems," IEEE Transactions on Computers, Vol.C-28, No.6, pp.395-406, 1979.

[DeWitt-86] DeWitt, D.J., Gerber, R.H., Graefe, G., Heytens, M.L., Kumar, K.B., and Muralikrishna, M., "GAMMA High Performance Dataflow Database Machine," Computer Sciences Technical Report #635, University of Wisconsin-Madison March 1986.

[DeWitt-87] DeWitt, D.J., et. al.,"A Single User Evaluation of the GAMMA Database Machine," Proc. of the 5th IWDM, pp.43-59, 1987.

[Federighi-94] Federighi, C. and Rowe, L.A., "A Distributed Hieradhical Storage Manager for a Video-on-Demand System," IS & T/SPIE,1994.

[Garcia-Molina-88] Garcia-Molina, H., and Salem, K., "The impact of disk striping on reliability," IEEE Database Engineering Bulletin, Vol..11, pp.26-39, Mar.1988.

[Gemmell-95] Gemmell, D.J., Vin, H.M., Kandlur, D.D., Rangan, P.V., and Rowe, L.A., "Multimedia Storage Servers:A Tutorial," Computer, May 1995.

[Gibson-89] Gibson, G.A., Hekkerstein, L., Karp, R.M., Katz, R.H., and Patterson, D.A., "Failure correction techniques for large disk arrays," in Proc. of the 3rd International Conference on Architectural Support for Programming Languages and Operating Systems, pp.123-32, Apr.1989.

[Gray-78] Gray, J.N., "Notes on Data Base Operating Systems," published in R.bayer, et.al. "Operating Systems:An Advanced-Course," Spriger-Verlag,1978.

[Hartman-93] Harman, J.H. and Ousterhout, J.K., "The Zebra Striped Network File System," Proceedings of the 14th ACM Symposium on Operating Systems Principles, 1993.

[IDM-85] The IDM 310 Database Server, Britton-Lee Inc.,1985.

[Kakuta-85] Kakuta T., Miyazaki, N., Sibayama, S., Yokota, H., and Murakami, K., "The Design and Implementation of the Relational Database Machine Delta," in Database Machines:Proc. of the 4th International Workshop, Springer Verlag, edited by D.DeWitt and H. Boral, March 1985.

[Kitsuregawa-83] Kitsuregawa, M., Tanaka, H., and Motooka, T.,"Architecture and Performance of Relational Algebra Machine Grace," University of Tokyo,Technical Report,1983.

[Lee-92] Lee, E.K., Chen, P.M., Hartman, J.H., Draperau, A.L., Miller, E.L., Katz, R.H., Gibson, G.A., and Patterson, D.A., "RAID-2:A Scalable Strage Architecture for High-Bandwidth Network File Service," Technical Report UCB/CSD 92/672, UCB,Feb.1992

[Long-94] Long, D.D.E., and Montague, B.R., "Swift/RAID:A Distributed RAID system," Computing Systems,vol.7,No.3,Summer,1994.

[Oracle] "Oracle Corporation response to the DeWitt Benchmark," Oracle Corp.

[Oyang-94] Oyang, Y.J., Lee, M.H., and Wen, C.H., "A Video Storage System for On-Demand Playback," Techinical report NTUCSIE 94-02, National Taiwan University, 1994.

[Ozden-95] Ozden, B., and Silberschatz, A., "A Framework for the Storage and Retrieval of Continuous Media Data," In Proc. of the IEEE International Conf. on Multimedia Computing and Systems, pp.2-13, May 1995.

[Ozkarahan-77] Ozkarahan, E.A., Schuster, S.A., and Sevcik, K.C., "Performance Evaliation of a Relational Associative Processor, ACM Trans. Database Syst., Vol.2, No.2, pp.175-195, 1977.

[Patterson-88] Patterson, D.A., Gibson, G., and Kotz, R.H., "A Case for Redundant Arrays of Inexpensive Disks(RAID)," Proc. of ACM SIGMOD,pp.109-116, 1988.

[Reddy-93] Reddy, A.L.N., and Wyllie, J., "Disk scheduling in a multimedia I/O system," Proc. of the 1st Intl. Conf. on Multimedia, Aug.1993.

[Rangan-91] Rangan, P.V., and Vin, H.M., "Designing File Systems for Digital Video and Audio," In Proc. of the 13th Symposium on Operating Systems Principles (SOSP'91), Operating Systems Review, Vol.25, No.5, pp.81-94, Oct.1991.

[Salem-84] Salem, K., and Garcia-Molina, H., "Disk striping," Technical Report No. 332, EECS Department, Princeton University, Dcemer 1984.

[Salem-86] Salem, K., and Garcia-Molina, H., "Disk striping," in Proc. of the 2nd International Conference on Data Engineering, pp.336-342, IEEE, Feb.1986.

[Simon-85] Simon, E., "Update to December 1983 'DeWitt' Benchmark," Britton Lee,Inc. 1985.

[Stonebraker-76] Stonebraker, M., Wong, E., and Kreps, P., "The Design and

Implimentation of INGRES," ACM Transactions on Database Systems, Vol.1,No.3, September,1976.

[Stonebraker-90] Stonebraker, M., and Schloss, G.A., "Distributed RAID-a new multiple copy algorithm," in Proc. of the 6th International Conference on Data Engineering, (Los Angeles), pp.430-437, IEEE,Computer Society, Feb.1990.

[Teradata-83] Teradata:DBC/1012 Data Base Computer Concept & Facilities, Teradata Corp. Document No..C02-0001-00,1983.

[Tobagi-93] Tobagi, F.A., Pang, J., Baird, R. and Gang, M., "Streaming RAID-A Disk Array ManagementSystem For Video Files," ACM Multimedia 93, Aug.1993.

[Ubell-85] Ubell, M., "The Inteligent Database Machine(IDM)," in Query Processing in Database Systems, edited by Kim, W., Reiner, D., and Batory,D., springer-Verlag,1985

[Ullman] Ullman, J., "Principle of Database Systems," Computer Science Press Inc.
（邦訳　日本コンピュータ協会版権，国井利泰訳"データベース・システムの原理"）

[Vin-94] Vin, H.M., Goyal, P., Goyal, A., and Goyal, A.,"A Statitical Admission Control Algorithm for Multimedia Servers," In Proc. of ACM Multimedia 94', pp.33-40, 1994.

[Vin-94] Vin, H.M., Goyal, A., Goyal, A., and Goyal, P., "An Observation-Based Approach For Designing Multimedia Servers," In Proc. of the IEEE Conf. on Multimedia Computing and Systems, pp.234-243, May 1994.

[Yu-92] Yu, P.S., Chen, M.S., and Kandlur, D.D., "Design and Analysis of a Grouped

Sweeping Scheme for Multimedia Storage Management," In Proc. of the 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, pp.38-49, Nov.1992.

[Yu-94] Yu, P.S., Chen, M.S., and Kandlur, D.D. "Grouped Sweeping Scheduling(GSS) for DASD Based Multimedia Storage Management," Research report, IBM T.J Watson Research center,1994.

[井上-90] 井上潮，速水治夫，福岡秀樹，鈴木健司，松永俊雄，"データベースプロセッサRINDAの設計と実現，" 情報処理学会論文誌，Vol.31. No.3. pp.373, 1990.

[岩見-93] 岩見直子，高原桂子，松井進，"LANにおけるマルチメディア通信方式の検討" 情報処理学会研究報告，マルチメディア通信と分散処理 61-1，1993.7.

[植村-77] 植村，弓場，古川，国分，大表，菅原，"磁気バブルによるデータベースマシンの構想，" 信学技報 EC76-78, 1977.

[植村-80] 植村俊亮，前川守，"データベースマシン，" 情報処理学会編,オーム社，1980.

[内川-94] 内川，横田，"VODにおける耐故障並列ディスクの利用とパケット不配／遅配への対応，" 情報処理学会研究報告 94-AVM-6, 情処研報 Vol.94,No.85, ISSN0919-6072. pp.1-7, 1994.10.

[岡-95] 岡，中西，池田，石田，清水，"NFSとTCP/IPを拡張し、動画に対応したビデオサーバ、MPEG1に相当の動画を途切れることなく配信，" 日経エレクトロニクス pp.133-141, 1995.9.25.

[喜連川-93] 喜連川優，"最近の二次記憶装置：ディスクアレイ，" 情報処理学会誌，Vol.34,

No.5, pp.642-651, 1993.5

[清木-87] 清木康，“データベースマシンの動向，”「アドバンスト・データベース・システム」システムシンポジウム論文集，pp.31-40, 1987.

[Kovalick-94] Kovalick, Al.，“ビデオサーバの設計手法、ディスク管理と高速バスが鍵、米国情報スーパーハイウェーを支える技術，”日経BP社 1994.10.

[坂本-93] 坂本泰久，桑名栄二，“TCP/IP上でのマルチメディア通信とその性能，”情報処理学会研究報告，マルチメディア通信と分散処理61-9，1993.7.

[阪本-95] 阪本秀樹，西村一敏，中野博隆，“ビデオ情報の大規模多重アクセス方式，”信学論（D-II），Vol.J78-D-II, no.1, pp.76-85,1995.

[阪本-96] 阪本秀樹，鈴木偉元，西村一敏，“ストライピング方式に基づいたビデオサーバにおけるディスク装置の負荷バランス解析，”信学論（D-II），Vol.J79-D-II, no.4, pp.634-638, 1996.

[清水-96] 清水洋，中村俊一郎，峯村治美，山口智久，渡辺尚，水野忠則，“分散RAID型V.O.D.におけるデータ配置問題について，”情報処理学会研究報告，マルチメディア通信と分散処理75-11, 1996.3

[田中-93] 田中裕之，平原正樹，荒木啓二郎，“インターネット上での音声会話ツール，”情報処理学会研究報告，マルチメディア通信と分散処理61-4，1993.7.

[高橋-85] 高橋義造，“計算機方式，”電子通信学会編，コロナ社 1985.7.

[堂之下-95] 堂之下謙二，堀内千尋，“簡易VOD構造の一検証，”情報処理学会第50回

全国大会 1995.

[速水-91] 速水治夫，“リレーショナル・データベースマシンにおける検索用人力バッファの最適構成，”情報処理学会論文誌，Vol.32. No.11. pp.1423, 1991.

筆者発表論文

[1] 坂和麿、村井真一、森村宏行、中村俊一郎、“MELCOM-COSMO700磁気ディスク制御装置”、昭和50年度電子通信学会全国大会、1975.

[2] 村井真一、中村俊一郎、“MELCOM/COSMO700磁気ディスク制御におけるマイクロプログラム”、電子科学10月号、1975.10.

[3] 横山繁盛、坂本、有賀幾夫、渡辺照久、中村俊一郎、“汎用電子計算機＜MELCOM EXシリーズ＞のハードウェアシステム”、三菱電機技報、Vol.59 No.7, 1985.7.

[4] 中村俊一郎、曽我正和、“高速データベースマシンＨＤＭのアーキテクチャ”、情報処理学会第35回全国大会、1987.9.

[5] 峯村治美、箕原辰夫、田口泰志、中村俊一郎、曽我正和、“高速データベースマシンＨＤＭの性能評価”、情報処理学会第35回全国大会、1987.9.

[6] 箕原辰夫、峯村治美、斉藤知人、中村俊一郎、曽我正和、“高速データベースマシンＨＤＭのアルゴリズム”、情報処理学会第35回全国大会、1987.9.

[7] 板倉國司、船橋正樹、武田浩良、中村俊一郎、曽我正和、“高速データベースマシンＨＤＭのソフトウェア”、情報処理学会第35回全国大会、1987.9.

[8] 花畑寿士、板倉國司、武田浩良、中村俊一郎、曽我正和、“高速データベースマシンＨＤＭのディスクアクセス方式”、情報処理学会第35回全国大会、1987.9.

[9] Shunichiro Nakamura, Harumi Minemura, Tatsuo Minohara, Kuniji Itakura, and Masakazu Soga, “A High Speed Database Machine HDM”, Proc. of the 5th International Workshop on Database Machines, pp.340-353, 1987.10.

[10] 箕原辰夫、峯村治美、中村俊一郎、石田喬也、“高速データベースマシンＨＤＭにおけるデータベースサーバ”、情報処理学会第36回全国大会、1988.3.

[11] 峯村治美、中村俊一郎、石田喬也、花畑寿士、“高速データベースマシンＨＤＭの性能評価（II）”、情報処理学会第36回全国大会、1988.3.

[12] 板倉國司、花畑寿士、中村俊一郎、石田喬也、“高速データベースマシンＨＤＭの集合アルゴリズム”、情報処理学会第36回全国大会、1988.3.

[13] 中村俊一郎、峯村治美、箕原辰夫、板倉國司、花畑寿士、“高速リレーショナルデータベースマシンＨＤＭ”、三菱電機技報、Vol.62 No.6, 1988.6.

[14] 峯村治美、板倉國司、下平純一、中村俊一郎、石田喬也、“高速データベースマシンＨＤＭの障害回復方式”、情報処理学会第 37 回全国大会、1988.9.

[15] 箕原辰夫、船橋正樹、中村俊一郎、石田喬也、“高速データベースマシンＨＤＭにおける Dynamic SQL の実現”、情報処理学会第 37 回全国大会、1988.9.

[16] 浅野拓也、花畑寿士、山内晋一、中村俊一郎、石田喬也、“高速データベースマシンＨＤＭにおける高速ホストインターフェースの開発”、情報処理学会第 37 回全国大会、1988.9.

[17] 浅野拓也、峯村治美、中村俊一郎、武藤達也、“高速データベースマシンＨＤＭのアーキテクチャ”、情報処理学会研究報告 89-ARC-78 情処研報 Vol.89 No.74, pp.1-8 1889.9

[18] 峯村治美、浅野拓也、佐藤誠、鹿島理華、中村俊一郎、武藤達也、“並列データベースマシンＨＤＭ”、電子情報通信学会研究報告 DE89-36-47 信学技報 Vol.89 No.335,pp.25-32, 1989.12.

[19] Harumi Minemura, Takuya Asano, Makoto Satoh, Rika Kashima, Hisashi Hanabata, Shunichiro Nakamura, and Tatsuya Mutoh, “International Conference on Database ahd Expert Systems Applications, pp.191-195, 1990.8.

[20] Shunichiro Nakamura, and Harumi Minemura “A Method of Fast Relational-Database Processing and Its Evaluation,” MITSUBISHI ELECTRIC ADVANCE , Vol.54,pp.26-28 ,1991.3.

[21] 喜連川優、金子悟、中村俊一郎、山本彰、辻澤隆彦、“パネル討論「ディスクアレイの現状と展望」”、情報処理学会研究報告 91-ARC-90 情処研報 Vol.91. No.86, pp79-80, 1991.10.

[22] 早川孝之、峯村治美、吉森幹夫、中村俊一郎、樋口雅弘、“RAID レベル 5 のためのバッファ管理方式とその性能評価”、電子情報通信学会研究報告 DE93-37-45 信学技報 Vol.93 No.251,pp.51-57, 1993.9.

[23] 峯村治美、中村俊一郎、吉村啓二、吉森幹夫、早川孝之、鹿島理華、“ミッドレンジビジネスコンピュータ用ディスクアレイのアーキテクチャ”、情報処理学会第 50 回全国大会,1995.3

[24] 小林剛、中島宏知、佐藤誠、植木則明、中村俊一郎、“ミッドレンジビジネスコンピュータ用ディスクアレイの高速化機構”、情報処理学会第 50 回全国大会、1995.3

[25] 山口智久、塩野勝美、峯村治美、中村俊一郎、“ミッドレンジビジネスコンピュータ用ディスクアレイの二重系制御方式”、情報処理学会第 50 回全国大会、1995.3

[26] 鈴木和雅、峯村治美、中村俊一郎、桑田圭三、小林清志、木岐将之、“ミッドレンジビジネスコンピュータ用ディスクアレイの高信頼化機能および性能”、情報処理学会第 50 回全国大会、1995.3

[27] 中村俊一郎、峯村治美、山口智久、清水洋、渡辺尚、水野忠則、“ビデオストリーム配信性能の一検証”、情報処理学会研究報告 マルチメディア通信と分散処理 72-7, 情処研報 Vol.95. No.85, 1995.9.

[28] 峯村治美、青砥久志、鹿島理華、吉森幹、中村俊一郎、“ソリューションサーバ用ディスクアレー”、三菱電機技報、Vol.69 No.10, 1995

[29] 中村俊一郎、峯村治美、山口智久、清水洋、渡辺尚、水野忠則、“分散 RAID 方式ビデオサーバー”、情報処理学会研究報告 マルチメディア通信と分散処理 73-22, 情処研報 Vol.95. No.115, 1995.12.

[30] 中村俊一郎、峯村治美、山口智久、清水洋、渡辺尚、水野忠則、“分散 RAID 方式ビデオサーバー(その 2)”、情報処理学会研究報告 マルチメディア通信と分散処理 74-39, グループウェア 15-39, 情処研報 Vol.96. No.12, 1995.12.

[31] 清水洋、中村俊一郎、峯村治美、山口智久、渡辺尚、水野忠則、“分散 RAID 型 V.O.D. におけるデータ配置問題について”、情報処理学会研究報告 マルチメディア通信と分散処理 75-11, 1996.3.

[32] 清水洋、中村俊一郎、峯村治美、山口智久、渡辺尚、水野忠則、“分散 RAID 型 V.O.D. におけるデータ配置問題について(その 2)”、情報処理学会研究報告 マルチメディア通信と分散処理 77-6, 1996.7.

[33] Shunichiro Nakamura, Harumi Minemura, Tomohisa Yamaguchi, Hiroshi Shimizu, Takashi Watanabe, and Tadanori Mizuno, "Distributed RAID Style Video Server", IEICE Transactions on Communications, pp.1030-1038, 1996.8.

[34] Shunichiro Nakamura, Harumi Minemura, Tomohisa Yamaguchi, Hiroshi Shimizu, Takashi Watanabe, and Tadanori Mizuno,"Multimedia Server on Ethernet", Proc.of International SymposiumCooperative Database Systems for Advanced Applications, pp.539-545, 1996.12.

[35] Shunichiro Nakamura, Harumi Minemura, Tomohisa Yamaguchi, Hiroshi Shimizu, Takashi Watanabe, and Tadanori Mizuno,"Some Experiments and an Implementation of Ethernet(10M/100M) Connected Video Server", Proc. of the 11th International conference on Information Networking, pp.5c-3.1-5c-3.8, 1997.1

[36] 中村俊一郎、峯村治美、山口智久、清水洋、渡辺尚、水野忠則、"イーサネット上のVOD システムの性能問題について"、静岡大学大学院電子科学研究科研究報告第 18 号（1996） pp.127-134, 1997.3.

[37] Shunichiro Nakamura, Harumi Minemura, Tomohisa Yamaguchi, Hiroshi Shimizu, Takashi Watanabe, and Tadanori Mizuno,"Poster on A Distributed RAID VOD System" Proc.of Second IFCIS International Conference on Cooperative Information Systems, pp.226, 1997.6.