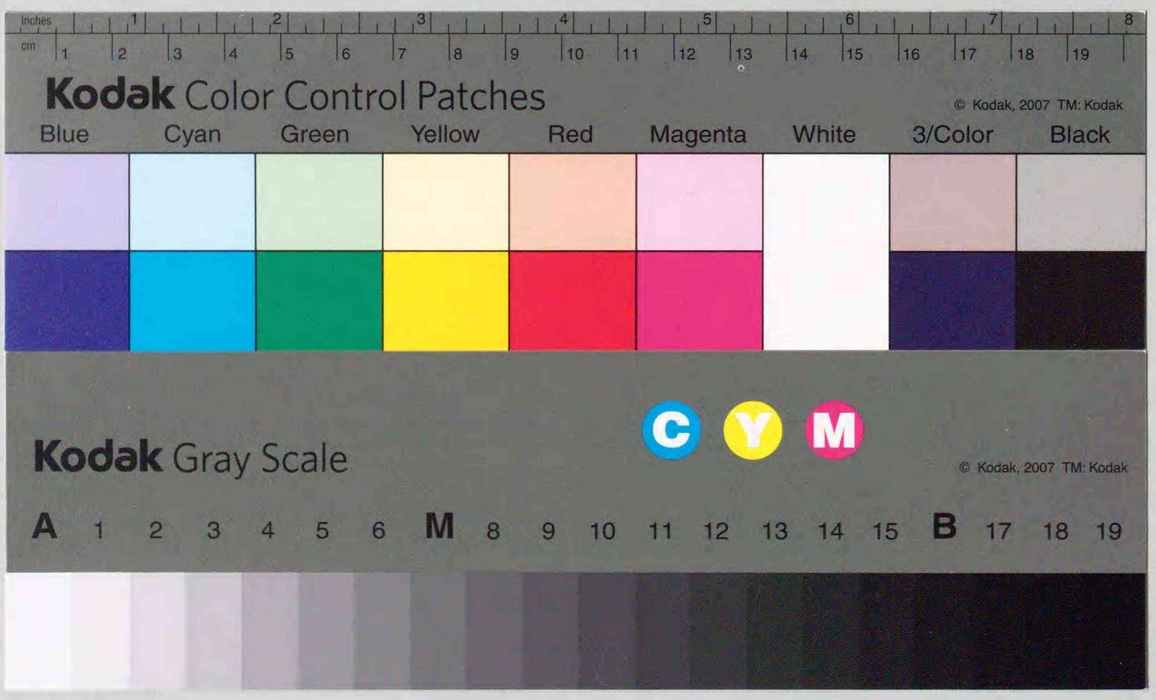
ディスク・サブシステムの
高スループット化に関する研究

平成5年 3月

岡田義広



ディスク･サブシステムの
高スループット化に関する研究

平成5年3月

岡田義広

目次

第1章 序論

本章では本研究の目的と成果の概要について述べる。また、本研究の背景を述べ他の研究との関連を明らかにする。

1.1節 本研究の目的と成果の概要

計算機システムは、主に中央処理装置、一次記憶装置、二次記憶装置(磁気ディスク装置)で構成される。磁気ディスク装置とそれを制御する制御装置を合わせて、ディスク･サブシステムと呼ぶ。一次記憶装置とディスク装置間のデータ転送は入出力チャネルを介して行われる。近年、ディスク･サブシステムは大容量化の傾向にある。磁気媒体を高密度化することにより磁気ディスク装置自体の記憶容量を大容量化する、あるいは、複数のディスク装置を設けることによりディスク･サブシステムを大容量化している。このような記憶容量の大容量化にも関わらず、磁気ディスク装置自体の処理速度の高速化は図られていないのが現状である。一方、ディスク･サブシステムがアクセスされる上位装置の処理性能は飛躍的に向上している。集積回路技術の進歩により中央処理装置自体を高速化する、あるいは、複数の処理装置を用いることにより処理性能の向上が図られている。したがって、ディスク･サブシステムは上位装置から頻繁にアクセスされる必要があり、ディスク･サブシステムの高スループット化が望まれる。本研究はディスク･サブシステムの高スループット化を目的とする。高スループットを達成するディスク･サブシステムのアーキテクチャを示し、その処理性能を明らかにすることを目的とする。

磁気ディスク装置は機械動作を含むため、ディスク･サブシステムの処理速度は中央処理装置、一次記憶装置の電子的処理速度に比べて遅い。中央処理装置、主記憶装置の処理速度とディスク装置の処理速度の差を緩和し、処理性能を向上させるためにディスクキャッシュが用いられる。ディスクキャッシュは、複数のパスを介して複数のディスク装置と接続される。さらに、複数のパスを介して入出力チャネルと接続される。ディスク･サブシステムを高スループット化するためには、ディスクキャッシュのデータ転送幅を上げる必要がある。複数の入出力ポートをもち、各ポートから独立にデータの入出力が行えるメモリをディスクキャッシュとして用いる必要がある。そこで、マルチポート･ページメモリ(MPPM : Multi-Port Page Memory)と呼ばれるメモリをディスクキャッシュとして用いることにした。マルチポート･ページメモリは、複数のメモリバンクと複

数の入出力ポートを内部接続網で接続した構成のメモリデバイスである。各ポートから独立にブロック単位でデータの入出力が行える。

すでに、MPPMをデータベース・マシンのディスクキャッシュとして用いる提案[1]があるが、汎用計算機に接続されるディスク・サブシステムのディスク・キャッシュとしてMPPMを用いる提案はない。特に、MPPMをディスクキャッシュとして階層的に用いる提案はない。さらに、MPPMをディスクキャッシュとして用いたディスク・サブシステムの種々のキャッシュ動作における処理性能に関する報告はない。本論文では、MPPMをディスクキャッシュとして階層的に用いたディスク・サブシステム(DIMP : DIsk subsystem with MPpm)のアーキテクチャを提案し、種々の構成とキャッシュ動作に関してその処理性能を明らかにする。MPPMをディスクキャッシュとして階層的に用いたディスク・サブシステムでは、MPPMを階層的に用いたことにより、構成上の自由度が高くシステム拡張が容易である。シミュレーションによる性能評価から、ディスク装置の処理性能あるいは入出力チャネルのデータ転送幅により制限されるまでの高いスループットが得られることが示された。この結果は論文(1)ですでに公表した。このシミュレーションによる性能評価は、待ち行列網に基ずく性能評価シミュレータCABを用いて行われた。このシミュレータは、本研究を遂行する上で不可欠である性能評価のためのツールとして開発された。このシミュレータに関する成果を論文(2)により公表した。また、MPPMをディスクキャッシュとして用いたディスク・サブシステムの二層全接続型構成と呼ぶ構成に関して、キャッシュデータのコヒレンシを考慮した4つのキャッシュ動作における処理性能を示し、それらの特性を明らかにした。キャッシュデータの置き換え方式として、ライトスルー(WTと略記)方式とライトバック(WBと略記)方式がある。二層全接続型構成と呼ぶ構成では、上層のMPPMと下層のMPPMのそれぞれに対してWT方式とWB方式を適用可能であるから、4通りのキャッシュ動作(WB/WB方式、WT/WT方式、WT/WB方式、WB/WT方式)が考えられる。これらのキャッシュ動作における性能を解析的に評価した結果から以下の特性が明らとなった。

限界スループットに関しては、WB/WB, WT/WB, WB/WT, WT/WT方式の順に性能が良くなる。低トランザクション域での平均応答時間に関しては、WB/WT, WT/WT, WT/WB, WB/WB方式の順に性能が良くなる。
この結果は論文(3)によりすでに公表した。さらに、信頼性を向上させるディスク構成であるディスクアレイに関して、MPPMをディスクキャッシュとして用いた場合の処理性能を示し、その特性を明らかにした。MPPMをディスクキャッシュとして用いたディスクアレイの処理性能は、既存のディスク・サブシステムのディスクアレイ性能の6~7倍程度の性能が得られることが示された。この結果は今後公表する予定である。

上記で述べたように、本研究の遂行にあたり性能評価は不可欠である。性能を評価する場合、実際に対象物を作り実験を行うことで厳密な評価性能が得られる。だが、研究レベルで考えられる種々のアーキテクチャすべてに関して、現物実験を行うことは非現実である。そこで、異なる構成と動作機構をもつ種々の対象システムの性能評価を効率良く行えるシミュレータが必要となる。種々の構成の対象システムについてそれらのモデリングが容易に行え、シミュレーション中の対象モデルの状態が容易に把握でき、対話的にシミュレーションが行える必要がある。このようなシミュレータを対話型グラフィカルシミュレータと呼ぶ。さらに、種々の動作機構をもつモデルをシミュレーション対象の構成要素として追加することにより、対象システムの複雑な動作をも記述可能となる。ディスク・サブシステムの動作を詳細にモデリングする場合には、種々の動作機構をもつ基本モデルが必要である。そこで、シミュレーション対象の構成要素の追加・登録が容易に行える必要がある。このような機能をもつシミュレーションシステムの開発を本研究の初期目的し、対話型グラフィカルシミュレータを開発するツール群を提供するシステムとしてFES(Flavor† Environment for Simulations)を開発した。さらに、このシステムを用いて待ち行列網を基本とした性能評価シミュレータ: CAB(Computer Architect's Board)を開発した。本論文では、FESが提供する種々の機能と特徴を述べ、開発した性能評価シミュレータ: CABにそれらの機能がどのように用いられているかを示す。また、ディスク・サブシステムの性能評価において、CABがどのように用いられるかを述べその有用性を示す。

FESは以下の機能を提供する。

(1) シミュレーションの対象モデルの構成を視覚的に定義する機能

- シミュレーションの対象モデルは基本構成要素(これを基本モデルと呼ぶ)の組み合わせにより定義される。基本モデルは画面上のグラフィックス・イメージとして視覚化される(これはダイアグラムのノードにあたる)。基本モデル間のデータの入出力関係を指定するためのモデルも視覚化される(これはダイアグラムのアークにあたる)。画面上でノードとアークを組合せ作図することにより、シミュレーションの対象モデルの定義が視覚的に行える。

(2) 対象モデルの状態や動作状況を視覚表示する機能

- 基本モデルの内部状態を任意のグラフィックス・イメージで表示できる。シミュレーション中の各基本モデルの動作状況がアニメーション表示され、対象モデルの動作を視覚的に把握できる。

(3) シミュレーション結果を種々の形式で表示する機能

- シミュレーション結果の妥当性の検証が行える。

(4) 対話型操作のための統合された環境

- 対象モデルの構成の定義、シミュレーション実行、対象モデルの妥当性の検証、シミュレーション結果の表示といった一連の操作を対話的に行える。

---

†Flavor : Lisp言語にオブジェクト指向を導入したオブジェクト指向プログラミング言語

これらは対話型グラフィカルシミュレータに必要な機能である。さらに、FESではMV(Model-View)モデリングと呼ぶモデリング手法を採用したことで、任意のオブジェクトを基本モデルとして追加できる。MVモデリングとは、画面上に表示される形状を定義した視覚表示部(View)と内部動作機構を定義した動作記述部(Model)に分けて基本モデルの定義を行うモデリング手法である。FESは種々の視覚表示部をあらかじめ用意している。動作記述部のみを定義し登録することで、任意の動作機構をもつオブジェクトを基本モデルとして追加できる。したがって、必要な動作機構をもつオブジェクトを新たに基本モデルとして追加することにより、複雑な動作の対象システムをもシミュレーション可能となる。待ち行列網を基本とした性能評価シミュレータCABでは、種々の動作機構をもつ基本モデルを提供することにより、ディスク・サブシステムの種々の構成における詳細な動作をも記述可能となった。

以上が本研究の目的と成果の概要である。

1.2節　本研究の背景(他研究との関連)

以下では、本研究の背景を述べ他の研究との関連を明らかにする。まず、対話型グラフィカルシミュレータの開発支援環境:FESと待ち行列網に基ずく性能評価シミュレータ:CABに関して他の研究との関連を述べる。その後、ディスク・サブシステムの高スループット化に関して背景を述べ他の研究との関連を述べる。

従来、計算機の入出力装置が扱えるデータはテキストのみであった。グラフィカルな入出力装置が採用されたのはグラフィックス・ワークステーションと呼ばれる計算機が登場してからである。それ以前のシミュレーションは、すべて言語レベルのシミュレータで行われていた。ハードウェア設計を目的として種々のシミュレーション言語が開発されている。それらには、DDL(Digital system Digital Language), CDL(Computer Design (description) Language), AHPL(A Hardware Programming Language)[2]などがある。これらはハードウェア記述言語(HDL:Hardware Description Language)と呼ばれる。これらのハードウェア記述言語によるシミュレーションでは、対象モデルの定義はすべてテキスチャルに記述(ソースプログラムを作成)する必要がある。専用のコンパイラにより、記述されたソースプログラムを実行形式へ変換し、実行プログラムを実行して結果を得るという過程を経る。モデリングが正しくない場合には、ソースプログラムを修正し上記の過程を繰り返す。また、異なる対象システムのシミュレーションについても、上記の過程を繰り返す必要がある。したがって、いくつかの異なる構成および動作の対象システムすべてに関してシミュレーションを行う場合には、上記の過程を幾度となく繰り返す必要があり、妥当な結果を得るまでに多大な時間を要する。そこで、対象モデルのモデリングを画面上で作図することにより視覚的に行える機能が必要となる。

対象モデルの視覚的な定義機能とシミュレーション結果の可視化機能をもつシステムとしてFLOWWARE[3]がある。これは、テキストを入出力とするハードウェア記述言語IDDAPのグラフィカルな入出力インターフェイスとして用いられる。IDDAPは、CDLのサブセットのシミュレータ言語である。作図による視覚的な対象モデルの定義により、アナログ回路シミュレータSPICEのソースプログラムを出力するシステムとしてM.G.Walkerらの開発したシステム[4]がある。また、同機能のシステムとしてSTEM[5]がある。これは、オブジェクト指向言語Smalltalk-80[6]上で、Smalltalk-80が提供するモデリング手法であるMVCを用いて開発されている。MVC(Model-View-Controller)モデリングとは、モデルの内部動作を記述した動作記述部(Model)と画面上の形状を定義した視覚表示部(View)と外部(ユーザ)からの操作に対する反応を定義した制御記述部(Controller)の3部分に分けてモデルを定義するモデリング手法である。彼らは、MVCモデリングを用いることにより、グラフィカルなインターフェイスが簡単に開発できることを示した。FESではMVモデリングと呼ぶモデリング手法を採用している。また、IC設計を対象としたシミュレータとしてPalladio[7]がある。これも、対象モデルの定義を図形入力により視覚的に行える機能をもつ。ただし、MARSと呼ばれるシミュレータの入力ソースプログラムを出力するもので、シミュレーションの実行はテキスト形式を入出力とするMARSが行う。以上で述べたシステムでは、図形入力形式で対象モデルの定義が視覚的に行える。ただ、シミュレーションの実行は、得られたソースプログラムを実行プログラムへ変換し、それを実行するという過程を経る。そのため、シミュレーション中の対象モデルの状態や動作の把握が行えない。

グラフィックス・ワークステーションと呼ばれるクラスの計算機が登場して、VLSI設計用のCADをはじめとしてグラフィカルな入出力が行えるシミュレータが多数開発された。それらには、INSIST[8], VEGAMES[9], ALHARD[10][11], PROCEED[12]などがある。また、論理回路理解のための教育用シミュレータとして小池田らの開発したシステム[13]がある。これらは、作図による視覚的な対象モデルの定義機能、シミュレーション中の対象モデルの状態および動作状況のグラフィカルな表示機能、シミュレーション結果の表示機能、およびシミュレーションのための一連の操作を対話的に行える統合環境を提供する。また、オブジェクト指向を導入し、VLSIの部品を視覚化されたオブジェクトとして定義し、データベース化するといったシステム[14]もある。これは図形入力によりIC回路を定義し、それを部品としてデータベースへ登録することで再利用しようというものである。登録された部品の利用時には、それらは再び画面上のグラフィックス・イメージとして表示される。同様の目的のシステムとして神戸[15]らの開発したシステムがある。これはVLSI設計のすべての処理を行える統合環境をもつ。

以上で述べた対話型グラフィカル･シミュレータはIC回路設計を対象としている。以下に、他の分野をシミュレーション対象とした対話型グラフィカル･シミュレータを挙げる。待ち行列網を基本としたグラフィカルなシミュレータとして、B.Melamedらの開発したシステム[16]がある。待ち行列モデルは画面上に視覚化される。それらを画面上で組合せ結線することにより、対象となる待ち行列網が定義される。各待ち行列モデルの待ち行列の長さも視覚化され、シミュレーション中には、それらが動的に変化し、対象システムのボトルネックが視覚的に把握できる。だが、このシステムでは異なる動作機構をもつ構成要素を新たに追加するという機能はない。マルチコンピュータ･システムの性能評価を目的に開発されたシミュレータとしてPARET[17]がある。ノードとアークを用いた図式入力により対象モデルの定義が行える。シミュレーション実行中の対象モデルの状態はノードの色により視覚的に把握できる。ペトリネットに基づいた動作記述が行えるシミュレータとしてG. Brunoらのシステム[18]がある。ペトリネットによって動作を記述し、それを基本構成要素として画面上で組合せ作図することにより対象モデルの定義が行える。シミュレーションは、ペトリネット･シミュレータが行う。したがって、シミュレーション中の動作の把握は行えない。また、分散コンピュータシステムの設計を目的としたシステムとしてSREM[19]がある。SREMはグラフィカルな入出力機能をもつ要求駆動型プログラミング言語である。文献[19]では、SREMを拡張して分散コンピュータシステムの設計が行えるようにした点を述べている。また、分散処理システム評価のためのシミュレータとしてSEDS[20]がある。これは、Formと呼ばれる書式を入力することで対象モデルの定義を行う。シミュレーションのための一連の処理を対話的に行える統合環境をもつ。

以上で挙げたシステムは、グラフィカルな入出力機能をもつシミュレータである。しかし、その対象は特定分野に固定されている。FESではMVモデリングを導入したことにより、種々の動作機構をもつオブジェクトをシミュレーション対象の基本構成要素として扱うことができる。オブジェクト指向を導入し、オブジェクト指向のメッセージ伝達機能によりシミュレーション実行が行えるシミュレーション用言語としてSIMMER[21]がある。これは、汎用に使えるシミュレーション言語であるがグラフィカルな入出力機能は提供されていない。また、シミュレーションが行えるようスプレッド･シートの機能を拡張したシステム[22]もある。セル間に制約を与え、その制約によってセル値の更新伝搬が行われる。各セル値をシミュレーション対象の構成要素の状態とみなすことができ、それが動的に変化するためシミュレーションを行うことができる。

プログラムを視覚的に定義する、あるいはプログラムの動作を視覚化するシステムもいくつか提案されている。プログラミングを図形の組合せにより視覚的に行う環境を提供するシステムとして、VennLisp[23],TinkerToy[24],Dialog.I[25],IDEOSY[26],PROGRAPH[27],PICT[28]等がある。入出力ポートを持つ種々の機能ブロックの各ポートを画面上で結線することでプログラムの定義を行うというシステム[29]がある。また、Smalltalk-80の環境において、グラフの構成の定義が作図により行えるようにしたシステム[30]がある。これは、ノードとアークを視覚化されたオブジェクトとして扱えるようにしたもので、MVCを拡張することにより実現されている。文献[30]では、このシステムの応用例としてペトリネットのシミュレータの例が示されている。図形をシンタックスに持つ関数型言語[31]も開発されている。プログラムを視覚化するものとして、T.Lehrらの開発したデバッガシステム[32]がある。また、市川らのシステム[33]がある。これはトレース行をプログラム中に入れ、トレース行に指定されたデータを視覚的に表示する機能を提供するものである。さらに、種々の機能を紙のイメージをもったオブジェクトとして定義し、それらの張り合わせにより機能の合成とレイアウト設計を行うシステム[34]も開発されている。

以上で挙げたシステムは、グラフィカルな入力機能をプログラミング時に提供する、あるいはグラフィカルな出力機能をプログラムの実行時に導入するものである。プログラミング言語は最も汎用なシミュレーション言語である。だが、これらにはシミュレーションの機能はない。したがって、対話型グラフィカルシミュレータに必要となる機能を全て作る必要がある。FESでは、対話型グラフィカルシミュレータに必要となる機能を提供し、さらに、任意の動作機構をもつオブジェクトをシミュレーション対象の構成要素(基本モデル)として利用できる機構を導入している。それがMVモデリングと呼ぶモデリング手法である。FESを用いて開発された待ち行列網シミュレータ:CABは、ディスク･サブシステムの動作機構をモデル化しシミュレーションするために必要となる基本モデルを容易に追加･登録できる。したがって、異なる構成と動作機構をもつ種々のディスク･サブシステムのシミュレーションも、それらに必要となる基本モデルを追加･登録することにより可能となる。さらに、種々の機能をもつVIEWを用意したことにより、種々の形態のモデリングが可能である。

以下では、ディスク･サブシステムの高スループット化に関して、背景と他研究との関連を述べる。

計算機システムは、主に、中央処理装置、一次記憶装置(主記憶装置)、二次記憶装置(磁気ディスク装置[35]など)で構成される。近年の集積回路技術の進歩により、中央処理装置、主記憶装置の処理速度は飛躍的に向上した。だが、磁気ディスク装置の処理速度の向上はほとんどみられない。磁気ディスク装置は機械動作を含むため、その処理速度は主記憶装置の電子的速度に比べて格段に遅い。主記憶装置の処理速度と磁気ディスク装置の処理速度の差は大きくなる一方である。その差は3オーダある。

ディスク装置の処理時間は3つの部分(シーク動作時間と回転待ち時間、データ転送時間)に分けられる。磁気媒体からデータを読み書きするための磁気ヘッド(あるいはリード/ライト･ヘッドという)を、該当するデータが含まれるトラック

まで移動するために要する時間をシーク動作時間という。シーク後、読み書きされるデータが磁気ヘッドの位置へ回転して来るのを待つ必要がある。これに要する時間が回転待ち時間である。その後、該当データが読み書きされる。読み書きに要する時間をデータ転送時間という。アクセスデータのサイズが小さな場合には、データ転送時間に比べてシーク動作時間、回転待ち時間が大きくなる。

近年、ディスク装置の記憶容量は大容量化の傾向にある。磁気媒体の記憶密度は年々上昇している[36]。磁気媒体の高密度化によりデータ転送時間は小さくなっている。だが、シーク動作時間、回転待ち時間は機械的動作時間であり、これらの技術的な向上はみられない。そこで、これらの時間を小さくしようという研究がある。R.E.Matickは、2つのリード/ライト･ヘッドを設け、外側のトラックと内側のトラックをそれぞれ担当させることによりシーク動作時間を短くするという機構[37]を考えた。また、連続してデータがアクセスされる場合には、シーク時間が短くなるようにそれらのアクセス順をスケジューリングしようという研究[38][39][40][41]がある。

ディスク装置の信頼性を上げる方式として二重書きがある[42]。2台のディスク装置を組みにして用い、データの書き込みを2台のディスク装置に対して行うことにより、常に同一の記憶情報が格納されるようにする方式である。2台のどちらか一方が故障しても、他の1台のディスク装置からデータの読み書きが行える。この方式のディスク装置をミラード･ディスクあるいはディスク･シャドーイングという。この方式のディスク装置におけるデータの読み込みは、2台のディスク装置の内、どちらか先に読み込み可能になったディスク装置に対して行われる。W. N. Spencerは、2台のディスク装置を同期して回転させ、それぞれのリード/ライト･ヘッドを1/2回転ずらした位置に設けることで、回転待ち時間を小さくするという考えを示した[43]。

磁気媒体の高密度化技術の向上により、小型ディスク装置の記憶容量も大容量となっている。しかも、汎用大型計算機で使われる大型ディスク装置の記憶容量/コスト比に比べて小型ディスク装置の記憶容量/コスト比のほうが優れていることが示されている[44]。低価格の小型ディスク装置装置を複数台用いて記憶容量を大きくし、しかも、それらを並列に動作させることでデータ転送幅を上げようという研究がある。これらのディスク装置をディスクアレイという。D.A.Pattersonは、ディスクアレイに少数台の冗長ディスクを付加することにより信頼性を上げるという提案[44]をした。冗長ディスクをチェックディスクと呼び、ユーザデータが格納されているディスクをデータディスクという。ディスクアレイ中の1台のディスク装置が故障した場合に、その故障ディスクに格納されていたデータを他の故障していないディスク装置に格納されているデータから回復する。これにより信頼性を上げるという方式である。任意のディスク装置が故障してもその格納データを回復できるようなデータ(これをチェックデータと呼ぶ)をチェックディスクに格納する。T. M. Olsonは、ランダムな入出力要求

におけるディスクアレイの性能評価[45]を行った。ディスクアレイに関しては、チェックデータのディスクへの格納方式やディスク構成を変えることでスループットや信頼性を向上させようという研究[46][47][48][49]がある。また、ディスクのアクセス単位となるブロックを分割し、これらを複数のディスク装置へ分けて格納し、複数のディスク装置から並行に読み書きすることによりデータの転送時間を小さくするという提案[50]がある。これをディスク･ストライピングあるいはディスク･インターリービングという。また、M.Y.Kimは、これらの複数のディスク装置を同期動作させることにより、制御が簡単になり性能も向上することを示した[51]。これらに関する性能評価もいくつ行われている[52][53][54]。ただし、これらではディスクキャッシュを含めた場合の性能については議論されていない。

以上は、ディスク装置自体の処理速度を向上しようというものである。中央処理装置、主記憶装置の処理速度とディスク装置の処理速度の差を緩和するための緩衝メモリを用いることにより、処理性能を向上させようという研究[55][56][57]がある。この緩衝メモリをディスクキャッシュという。

ディスク装置から読み出される確率の高いデータをあらかじめディスクキャッシュへ格納しておく。ディスク装置からの読み出し要求に対して、その要求データがキャッシュ内にある場合、ディスク装置をアクセスせず、キャッシュから即座にデータの読み出しが行われる。これにより、ディスク装置に対するアクセス回数を減らすことができ高速に処理が行える。しかし、キャッシュ内に該当データがない場合、ディスク装置をアクセスする必要がある。該当データがキャッシュ内に存在しない時をキャッシュミスという。逆に、該当データがキャッシュ内に存在する時をキャッシュヒットという。キャッシュヒットの割合(これをキャッシュのヒット率という)が高いほど処理速度が向上する。だが、ヒット率が低い場合には性能の向上はない。しかも、スループットはディスクキャッシュを用いない時より低下する。それは、キャッシュのヒット率を上げるため、ディスク装置をアクセスする時に連続したデータ(数ブロックから数トラック)を一度に読み込むからである。ヒット率が低い場合には、このデータ転送は無駄となる。キャッシュのヒット率が低い場合にも高スループットが得られる必要がある。そのためには、ディスク装置からディスクキャッシュへのデータ転送を効率良くし、ディスク装置とディスクキャッシュ間パスの限られた転送幅を有効に利用する必要がある。ディスクキャッシュの性能に関するいくつかの研究[58][59][60][61][62]がある。

従来のキャッシュ付きディスク･サブシステムでは、複数のディスク装置が数本のパスでディスクキャッシュと接続された構成で、そのパスを介してデータ転送を行う。ディスク装置がシーク動作と回転待ちをしデータ転送が可能になったにも関わらず、これらのパスが他のディスク装置のデータ転送で占有されている場合には、このディスク装置はデータ転送が行えない。この場合には、こ

のディスク装置はもう1回転待つ必要がある。これをRPS(Rotational Positioning Sensing)ミスという。転送パスが使用可能になるまで1回転待ちを繰り返す必要がある。ディスク装置内部に小容量のバッファメモリを設け、回転待ち時間を小さくしようという研究[63][64]がある。転送パスが使用できない時には、このバッファメモリへ一旦データを格納する。転送パスが使用可能になった時点で、このバッファメモリからをデータの転送を行う。ディス装置の回転とは非同期にデータ転送が行え、1回転待ちによる無駄時間を小さくでき処理速度が向上する。だが、スループットの飛躍的な向上は見られない。

ディスク装置からディスクキャッシュへ転送する複数ブロックを過去のアクセスの履歴情報により決定するという方法[65]がある。この方法では、ディスク装置とディスクキャッシュ間のデータ転送パスの有効利用が図られ、ディスクキャッシュのヒット率が低い場合にも高性能が得られる。だが、ディスク･サブシステムにおいて、ディスク装置の処理速度によって制限されるほど高いスループットを得るためには、ディスクキャッシュとディスク装置間のデータ転送パス幅を大きくし、さらに、ディスクキャッシュと入出力チャネル間のデータ転送パス幅をも大きくする必要がある。そのためには、ディスクキャッシュのデータ転送幅を上げる必要がある。

従来のキャッシュメモリ構成では、高速なメモリモジュールとバッファを用いて仮想的に多重ポート化している。この構成では、ポート数を増やす場合、さらに高速なメモリデバイスを用い、高速にポート切り替えの制御を行う必要がある。特に、数十ものポート数が必要な場合には実現が不可能である。そこで、何らかの工夫が必要となる。Y.Tanakaは、マルチポート･ページメモリと呼ばれるメモリを提案し、ディスクキャッシュとして用いることを提案[1]した。マルチポート･ページメモリは、複数のメモリバンクと入出力ポートを内部接続網で接続した構成のメモリデバイスである。各ポートから独立に、ブロック単位でデータのアクセスが行える。ディスクキャッシュとしてマルチポート･ページメモリを用いることにより、小さなオーバヘッドで、データ転送幅を増やすことができる。また、マルチポート･ページメモリをバッファメモリとして用いた知識ベースマシンの研究[66]もある。だが、これらでは、マルチポート･ページメモリを一般のディスク･サブシステムのディスクキャッシュとして用いた場合のキャッシュ動作や処理性能については議論していない。また、マルチポート･ページメモリを階層的に用いた場合のキャッシュ動作や処理性能についても議論していない。

本論文では、マルチポート･ページメモリをディスクキャッシュとして用いることにより、ディスク装置の処理性能により制限されるほど高いスループットが得られることを明らかにした。また、マルチポート･ページメモリを階層的に用いることを提案し、キャッシュデータのコヒレンシを考慮した4つのキャッシュ動作と処理性能を示し、それらの特性を明らかにした。さらに、マルチポート･ページメモリをディスクキャッシュとして用いた場合のディスクアレイ･システムにおける処理性能を明らかにした。

1.3節　本論文の構成

以下では、まず第2章で、本研究の初期目的である対話型グラフィカルシミュレータの開発支援環境:FESとFESを用いて開発された待ち行列網シミュレータ：CABについて述べる。FESの設計思想として、MVモデリングについて述べる。FESの特徴とFESが提供する種々の機能について解説する。FESが提供する種々の機能にMVモデリングがどのように用いられているかを述べる。さらに、FESが提供する機能が、CABの開発とその特徴にどのように現れているかを述べる。また、実際のシミュレーション例を挙げ、種々の異なる構成と動作機構をもつ対象システムのシミュレーション実験が、CABを用いることにより少ない作業量で容易に行えることを述べる。第3章からは、本研究の主目的であるディスク･サブシステムの高スループット化について述べる。第3章では、マルチポート･ページメモリ(MPPM)をディスクキャッシュとして階層的に用いたディスク･サブシステム(DIMP)のアーキテクチャについて述べる。まず、従来のキャッシュ付きディスク･サブシステムのボトルネックを解説する。シミュレーションにより得られた、従来のキャッシュ付きディスク･サブシステムの処理性能とDIMPの処理性能を示し比較を行う。また、ディスク･サブシステムの性能評価においてCABがどのように用いられているかを示し、CABの有用性を述べる。DIMPでは、入出力チャネルの転送幅によって制限される、あるいは、ディスク装置の処理性能により制限されるまでの高いスループットが得られることを示す。第4章では、DIMPの二層全接続型構成と呼ぶ構成について、キャッシュデータのコヒレンシを考慮した4つのキャッシュ動作とその処理性能を示す。M/M/1待ち行列モデルを用いて解析的に得た処理性能より、それらの4つのキャッシュ動作の特徴を述べる。第5章では、マルチポート･ページメモリをディスクキャッシュとして用いたディスク･サブシステムのディスクアレイについて、そのキャッシュ動作と処理性能を示す。従来のキャッシュ付きディスク･サブシステムとの性能比較を行う。ディスクアレイでは、冗長ディスクに対するアクセスが増え、ディスクアレイでない場合に比べてスループットが低下する。マルチポート･ページメモリをディスクキャッシュとして用いたディスクアレイ･システムでは、既存のディスク･サブシステムよりも高いスループットが得られることを示す。最後に、第6章で本論文の結論を述べる。各章を総括し、本研究の成果の要約と今後の課題について述べる。

第2章 対話型グラフィカルシミュレータの開発支援環境：FES(Flavor Environment for Simulations)

本章では、1章で述べたように、本研究の初期目的である対話型グラフィカルシミュレータの開発支援環境：FESについて述べる。FESの特徴とFESが提供する種々の機能について解説する。また、FESを用いて開発された待ち行列網を基本とした性能評価シミュレータ：CABを示し、その開発でFESが提供する機能がどのように用いられているかを述べる。また、異なる構成と動作機構をもつ種々のディスク･サブシステムの性能評価において、CABの特徴と機能がどのように用いられているかを述べ、その有用性を示す。FESおよびFESを用いて開発されたシミュレータは、プログラミング言語Common Lisp[67]およびオブジェクト指向言語Flavor[68]を用いて開発されている。図2.1に示されるのは、FESを用いて開発されたシミュレータCABの画面ハードコピーである。このウィンドウ内でシミュレーション対象のモデル化、シミュレーションの実行、シミュレーション中の対象モデルの各構成要素の状態･動作の確認、シミュレーション結果のグラフ表示などがマウスを用いて対話的に行える。

以下では、2.1節でFESの設計思想を述べる。ここで、MVモデリングについて説明する。2.2節でFESのシステム構成について説明する。ここで、FESが提供する種々の機能を解説する。2.3節でFESの応用システムとして開発した待ち行列網シミュレータCABを挙げてFESの種々の機能がどのように用いられているかを述べる。また、CABを用いて性能評価した例を示しCABの有用性を述べる。FESを用いて開発された対話型グラフィカルシミュレータとして、CABの他に2つのシミュレータがある。ペトリネットシミュレータPABと論理回路シミュレータLABである。これらについても簡単に説明を加え、FESの有用性を示す。2.4節で本章のまとめを述べる。

2.1節　FESの設計思想

本節ではFESの設計思想について述べる。シミュレーションによって計算機システムの性能評価を行う場合に一般にどのような過程で行われるかを述べ、それらの過程に要する作業量を減らすためにはどのような機能が必要かを示す。FESがそれらの機能をどのような形式で提供しているかを述べる。

(1)　プログラミング言語による性能評価

一般に、プログラミング言語あるいはシミュレーション用言語による性能評価では以下に挙げる手続を行う。

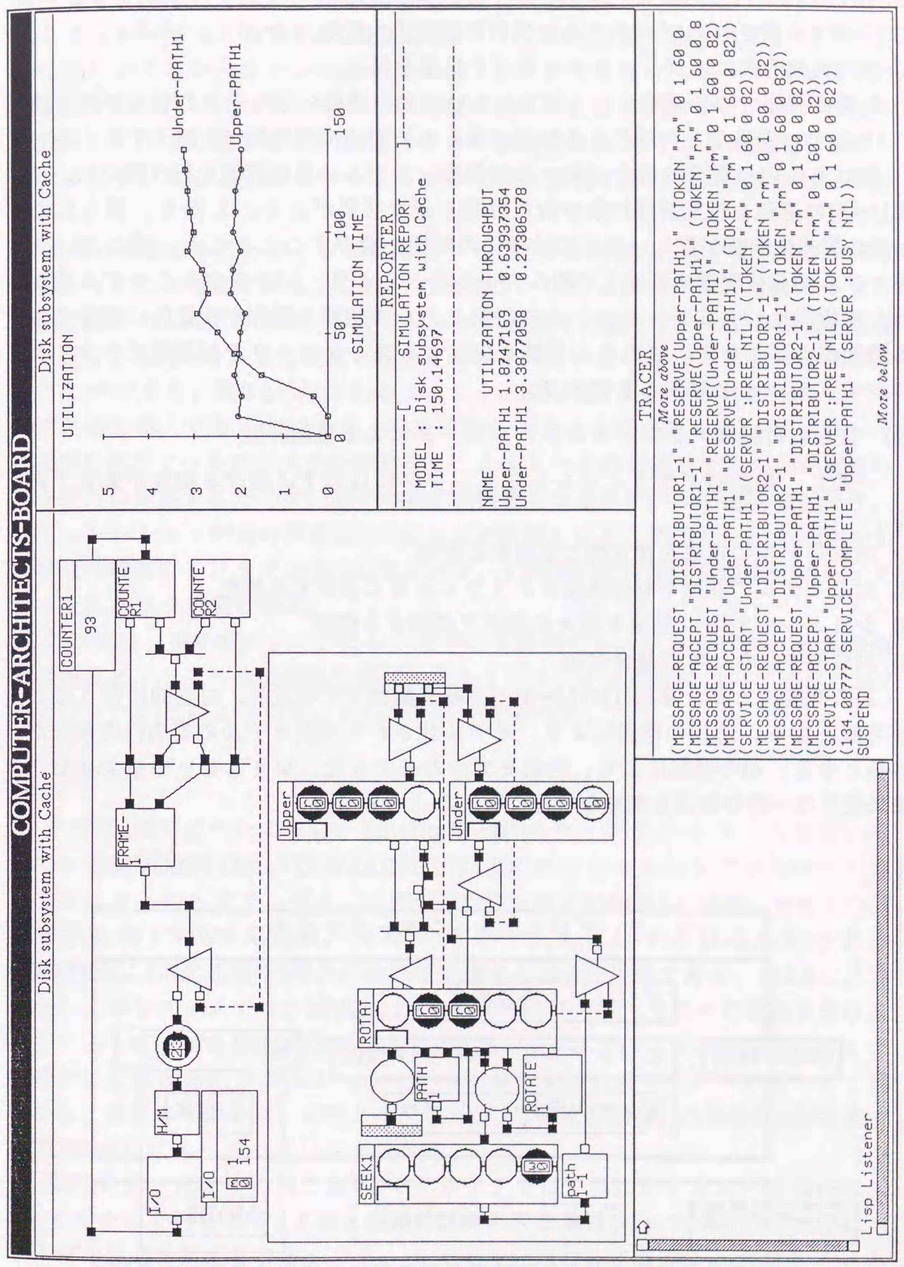


図2.1　FESが提供するSimulation-Windowの画面ハードコピー

1) 性能評価対象のモデル化を行う。
2) モデル化された対象モデルをプログラミング言語などを用いて定義する。
3) 上記で定義されたプログラムを実行可能形式へ変換(コンパイル)する。
4) 実行可能形式のオブジェクトを実行し結果を得る。
5) 結果が妥当でない場合には上記1)あるいは2)の過程へ戻りそれ以後の手続を繰り返す。妥当だと判断される結果が得られた時点で性能評価を終了する。

構成あるいは動作の異なる種々の計算機システムの性能評価を行う際には、上記1)~5)の過程を性能評価対象の数だけ繰り返す必要がある。しかも、得られた結果の妥当性判断が難しい場合には多くの時間を費やすこととなる。そこで、対象モデルの構成要素を画面上で組み合せることにより、対象モデルのモデル化が行える機能やシミュレーション中の対象モデルの状態や動作が視覚的に把握できる機能が必要となる。これらの機能を提供するシミュレータを対話型グラフィカルシミュレータと呼ぶことにする。

(2) 従来の対話型グラフィカルシミュレータによる性能評価

一般に、対話型グラフィカルシミュレータには以下に挙げる機能が必要である。

1) 対象モデルの構成を視覚的に定義する機能
2) 対象モデルの状態や動作状況をグラフィカルに表示する機能
3) シミュレーション結果を種々の形式で表示する機能
4) 対話型操作のための統合環境

1)の機能により、本節(1)の1)~3)の手続を画面上で視覚的、対話的に行うことが可能となる。2),3)の機能により、モデル化された対象モデルの妥当性の検証が容易となる。4)の機能により、対象モデルのモデル化、シミュレーション実行、結果表示の一連の作業を簡単な操作で行えるようになる。

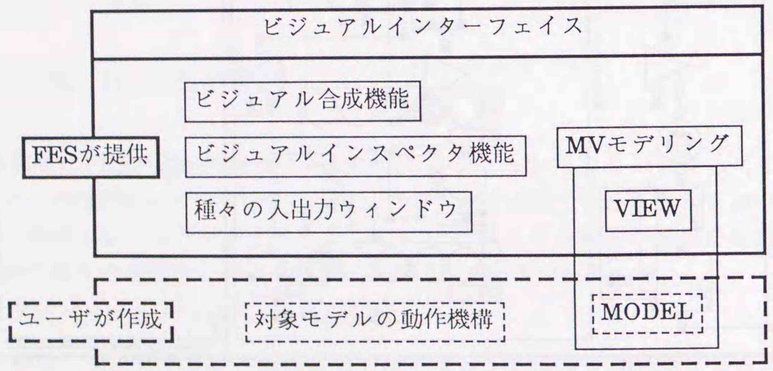


図 2.2 FESが提供するビジュアル機能

1章で述べたように、VLSI設計用のCADをはじめとして、上記の機能をもつグラフィカルなシミュレータは数多く開発され提案されている。だが、これらはシミュレーション対象がVLSI設計用といった特定分野に限られる。種々の分野に渡りシミュレーション対象として扱うことができるシミュレータは著者の知る限りない。シミュレーションの対象が種々の分野に渡る場合には、それらの分野のすべての基本モデル(対象モデルの構成要素)をあらかじめ定義し用意しておく必要がある。そのようなシミュレータを開発することは現実には不可能である。

上記の機能はシミュレーションを行う上で必要な入出力作業を視覚的かつ直接的に行うための機能である。これらの機能は、シミュレーション対象のモデルの動作機構には依存しない。シミュレータの機能を対象モデルの動作機構を定義する部分と視覚的な入出力機能であるビジュアルインターフェイス部分とに分けることができる。図2.2に示されるように、FESはMVモデリングと呼ぶモデリング手法を導入することにより上記の二者を分け、ビジュアルインターフェイス部分を汎用ツールの形式で提供する。シミュレータの対象モデルの動作機構のみを新たに定義するだけで、種々の動作機構をもつ基本モデルを追加·登録でき、シミュレーション対象の詳細な動作をも記述可能となる。FESでは、上記(2)の1)に挙げた機能をビジュアル合成機能と呼び、上記(2)の2)に挙げた機能をビジュアルインスペクタ機能と呼ぶ。

以下では、MVモデリングについて詳しく説明する。また、本システムの特徴であるビジュアル合成機能とビジュアルインスペクタ機能について述べ、MVモデリングがどのように用いられているかを述べる。

2.1.1 MVモデリング

すでに1章で述べたように、Smalltalk-80[6]のウィンドウ·システムに用いられているMVC(MODEL,VIEW,CONTROLLER)によるモデリング手法は、対象オブジェクトのモデリングを、内部動作機構を表すMODEL記述部、画面上の表示形状を表すVIEW記述部、マウスやキー入力に対する反応を定義するCONTROLLER記述部の3部分に分けて定義する抽象化手法である。図2.3に示すように、本システムでは、MVCにおけるCの機能を固定しVにその機能を含めることによって、MとVの2部分による記述手法を採用した。これをMVモデリングと呼ぶことにする。シミュレーションの対象モデルは基本モデルの組合せで構成され、各基本モデルは、画面上の視覚的形状を表すVIEWと内部動作機構を表すMODELの対によってモデリングされる。

図2.3に示されるように、MVCモデリングでは、Mに対するユーザ操作は、主にCを介して行われる。C部を他のCに換えることによって、ユーザ操作に対するMの反応を換えることができ、モデルの記述力を高めることができる。しかし、MとVとCの3部分を常に記述しなければならなく、モデル定義の際の記述

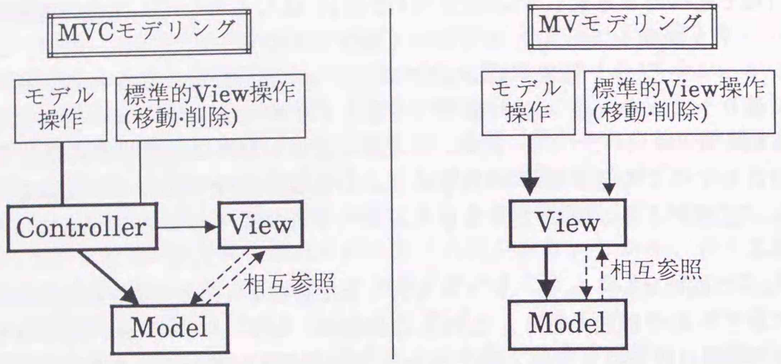


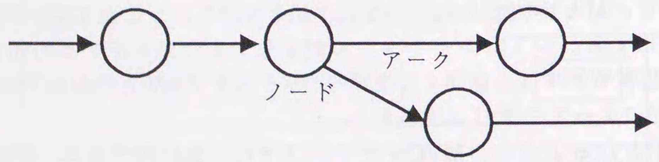
図2.3 MVCモデリングとMVモデリング

量が増える。Cの機能が限定されており、Mに対するユーザ操作が常にVを介して行われるならば、CをVに含めることができ、MとVの2部分によるモデリング手法を用いることができる。

本システムでは、基本モデルに対するユーザ操作をMODELに対するもの(保持する属性値や結果の表示および変更)と、VIEWに対するもの(移動や複写·削除など)とに分けた。VIEWはディスプレイ画面上のグラフィックス·イメージとして視覚化されており、ユーザ操作はすべて画面上で、VIEWを介して行うものとした。ユーザ操作が、移動や複写などのVIEWに対するものであればVIEWがそれを処理する。属性値の表示や変更などのMODELに対するものであれば、VIEWからMODELへ対応するメッセージが送られ、送られたメッセージにしたがってMODELがそれを処理する。このように、VIEWに対する操作とMODELに対する操作を区別しその種類を限定することで、VIEWをあらかじめ定義·登録しておくことができる。

2.1.2 ビジュアル合成機能

ビジュアル合成機能とは、ディスプレイ画面上に基本モデルの実体が存在するかのごとく、移動や複写·削除などの操作を視覚的に行える環境を提供し、ディスプレイ画面上で作図することにより、対象モデルの構成の定義が行える機能である。これは、ビジュアル·プログラミング[69]の手法であるダイアグラマティック·プログラミングやアイコニック·プログラミングで用いられるインターフェイスと同様の機能を実現したものである。



ノード:対象モデルの基本構成要素(基本モデル)を表す
アーク:基本モデル間の接続(データの入出力)関係を表す

図 2.4 ダイアグラム

MVモデリングでは、VIEWはMODELの定義には依存せず、あらかじめ定義·登録しておくことで汎用に使えるものである。そこで、図2.4に示されるようなダイアグラムのノードとアークにあたるVIEW(後述するPorts-VIEWとLine-VIEW)を用意した。これにより、ダイアグラムによる図表現によってその構成が記述できる事物であればシミュレーション対象として扱うことができる。ディスク·サブシステムなどの計算機システムをシミュレーション対象とする場合、対象システムは基本構成要素に分解できる。基本構成要素の動作機構を個々に定義し、それら基本構成要素間のデータの入出力関係を定義することにより、シミュレーション対象とする計算機システムのモデリングが行える。基本構成要素をノードとし、それら基本構成要素間のデータの入出力関係を表すものをアークとすることで、ダイアグラムの図表現により計算機システムの構成の定義が行える。

2.1.3 ビジュアルインスペクタ機能

従来の対話型グラフィカル·シミュレータでは、トレース機能などによりシミュレーション中の対象システムの状態の把握が行える。だが、トレース機能では状態を数値や文字列として表示するだけで、状態の把握がし易いとは言えない。ビジュアルインスペクタ機能とは、基本モデルの詳細な状態をディスプレイ画面上のグラフィックス·イメージとして視覚表示するための機能である。この機能によりシミュレーション中の対象システムの状態の把握が容易に行え、対象システムのモデリングの妥当性検証が行える。上記の基本モデルの視覚表示部Ports-VIEWは、後述するように、対象システムの構成を定義する場合に用いるアイコンでしかない。そこで、基本モデルの状態をグラフィックス·イメージとして視覚表示するための専用の視覚表示部(後述するDetail-VIEW)を設けた。

一般に、グラフィカルな表示機能を実現する場合、そのための処理によるオーバヘッドが生じ、全体の処理速度を低下させる。ビジュアルインスペクタ機能では、Detail-VIEWの表示、非表示を指定することができる。Detail-VIEWが表示されている場合には、対応する基本モデルの状態や動作がグラフィカルに表示される。Detail-VIEWが表示されていない場合にはグラフィカルな表示はされず、シミュレーションの実行速度の低下を抑えることができる。

2.2節　FESのシステム構成

FESは、シミュレーションを対話的に行うための統合環境を提供するウィンドウ·システムと視覚的な入出力機能を提供するMVシステムで構成されている。また、イベント駆動方式で動作するシミュレーション·ドライバを用意している。以下、それぞれに関して詳しく説明する。

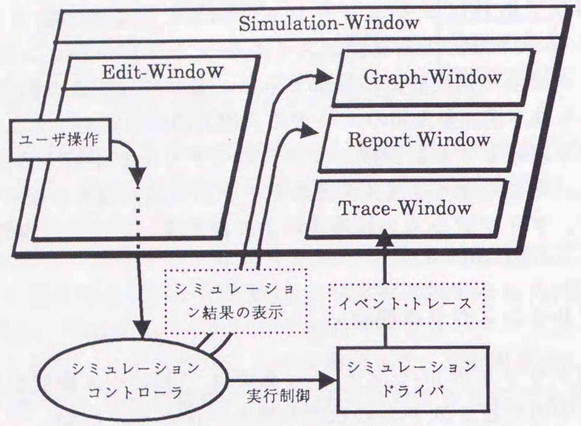


図 2.5　ウィンドウ間の関係

2.2.1　ウィンドウ·システム

FESでは、対象モデルの構成の定義、シミュレーション結果のグラフ表示、レポート作成、イベント·トレースの各機能に対応して、以下に挙げる4種類のウィンドウがシステム·ウィンドウ(Simulation-Window)のサブ·ウィンドウとして用意してある。図2.5に示されるようなシミュレーションのための統合環境を提供する。

(1) Edit-Window : このウィンドウ内でシミュレーションの対象モデルの定義を行う。基本モデルの視覚表示部であるVIEW(Ports-VIEWとLine-VIEW)を画面上で組合せることによりシミュレーションの対象モデルの定義が行える。また、シミュレーションの実行や結果の表示などの指示はこのウィンドウ内で行う。

図2.6に示されるように、VIEWはEdit-Window内に存在し、Edit-Window内でのマウスやキー入力は一度Edit-Windowのプロセスに渡される。この入力がEdit-Window上の特定のVIEWに対するものであれば、Edit-Windowはこの入力メッセージをそのVIEWに渡す。各VIEWには一つのMODELが割り当てられており、VIEWに送られた入力メッセージがそのMODELによって解釈されるもの

であれば、このメッセージはMODELに渡される。MODELとVIEWは互いに他を参照でき、メッセージを送ったり必要なデータを渡すことができる。

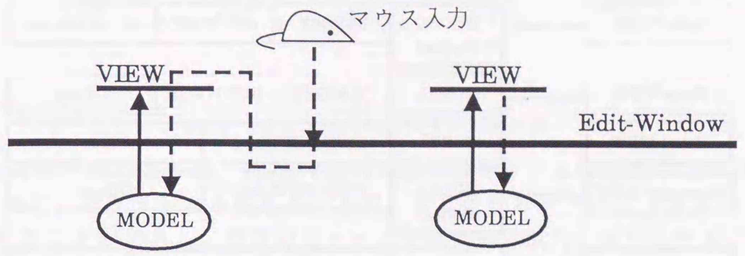


図2.6　Edit-WindowとVIEW, MODELの関係

(2) Graph-Window : シミュレーション実行によって得られた種々の数値結果をグラフ表示するためのウィンドウである。

(3) Report-Window : 各種数値結果を任意の書式にフォーマットして表示するためのウィンドウである。

(4) Trace-Window : イベント駆動によって動作しているシミュレータのイベント生起過程をトレースし保存するためのウィンドウである。このトレースデータを確認することで対象システムの動作を把握でき、対象システムのモデル化の妥当性検証が行える。

これらのウィンドウでは、すべての操作をマウスを用いて行え、対象システムの構成の定義、シミュレーションの実行、結果表示の一連の処理を対話的に行うことができる。先に示した図2.1が、Simulation-Windowの画面ハード·コピーである。左上がEdit-Window、右上がGraph-Window、右中がReport-Window、そして、右下がTrace-Windowである。

2.2.2　MVシステム

MVシステムは、MVモデリングおよびビジュアル合成機能、ビジュアルインスペクタ機能を提供するものである。ビジュアル合成機能、ビジュアルインスペクタ機能を提供するために、以下に示す4種類のVIEWがあらかじめ用意されている。以下それぞれのVIEWについて解説する。これらのVIEWが持つスロット(インスタンス変数)の種類とMODELが持つスロットの種類を図2.7に示す。図中において使われているスロット名とクラス名の枠線の種類は、そのスロットに対応するオブジェクトがどのクラスに属するかを示している。

(1) Ports-VIEW & Line-VIEW : 図2.8左にグラフィックス·イメージが示されているように、それぞれダイアグラムのノードとアークにあたるVIEWである。Ports-VIEWは任意の数の入出力ポートを持ち、Line-VIEWは一対の入出力ポートを持つ。白抜きの小さな四角形が入力ポートであり、黒く塗りつぶされた小さな四角形が出力ポートである。Ports-VIEWとLine-VIEWの入出力ポートのそれ

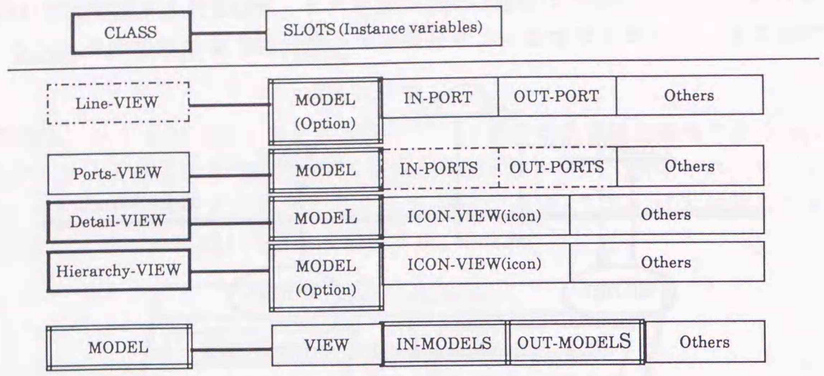


図 2.7　VIEWとMODELが持つスロットの例

ぞれを画面上で接続することで、ダイアグラムを構成できる。図2.8右は、図2.8左の構成の一部について、スロットの参照関係を示したものである。これに示されるように、Line-VIEWのスロットIN-PORT、OUT-PORTには、入出力ポートに接続されているPorts-VIEWがそれぞれ格納される。同様に、Ports-VIEWのスロットIN-PORTS、OUT-PORTSには、入出力ポートに接続されている数個のLine-VIEWがリストとしてそれぞれ格納される。Line-VIEWによって接続された二つのPorts-VIEW間の入出力関係は、それらのPorts-VIEWに割当てられているMODEL間の入出力関係となる。Ports-VIEWは、割当てられているMODELを参照するためのスロットMODELを持つ。同様に、MODELは、VIEWを参照するためのスロットVIEWを持ち、スロットIN-MODELSとOUT-MODELSには、そのMODELの入力側、出力側のMODELがそれぞれリストとして格納される。

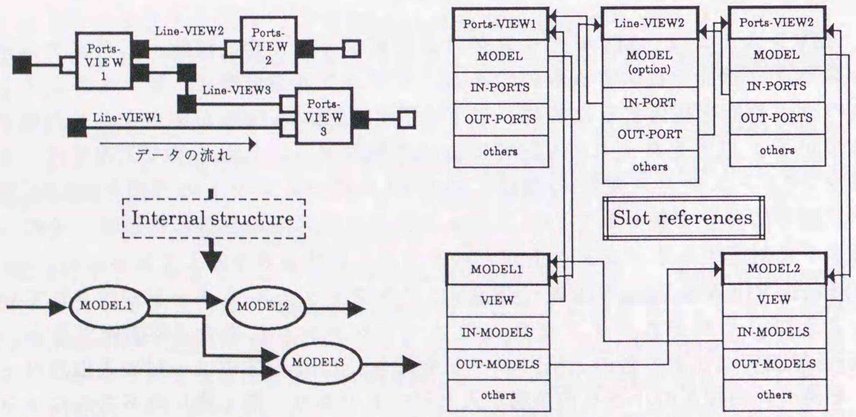


図2.8　Ports-VIEWとLine-VIEWの接続関係とそれらの内部構造とスロット参照関係

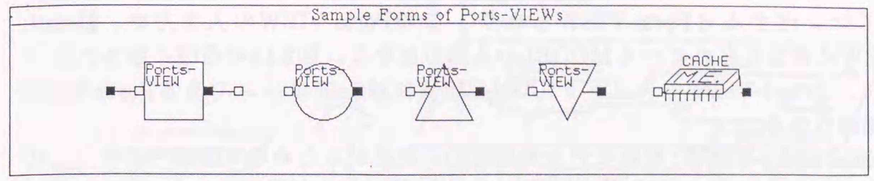


図 2.9　Ports-VIEWが持つフォームの例

図2.9に示されるように、Ports-VIEWは種々のフォーム(長方形、三角形、円など)を持ち、パラメータを変えることで、表示されるグラフィックス･イメージを変えることができる。任意のフォームを新たに作成し、表示させることも可能である。対応するMODELの種類により、表示されるフォームを変えることで、そのVIEWに対応するMODELが何であるかを視覚的に知ることができる。また、MODELからの指示にしたがって表示されるフォームを切り替えることもできる。MODELの状態変化にしたがってVIEWに表示されるフォームを変えることで、視覚的にモデルの状態が把握できる。ただし、表示されるフォームの大きさはVIEWの大きさに制限される。任意の大きさのフォームを表示したい場合には次に説明するDetail-VIEWを用いる。MODELが割当てられていないPorts-VIEWは、以下に述べるDetail-VIEWやHierarchy-VIEWのアイコンとして用いられる。

(2) Detail-VIEW : これはMODELの状態をグラフィカルに表示するためのVIEWである。このVIEWはディスプレイ画面上にグラフィカルな表示が行えるような任意の大きさの領域を提供するもので、表示されるグラフィクイメージは、

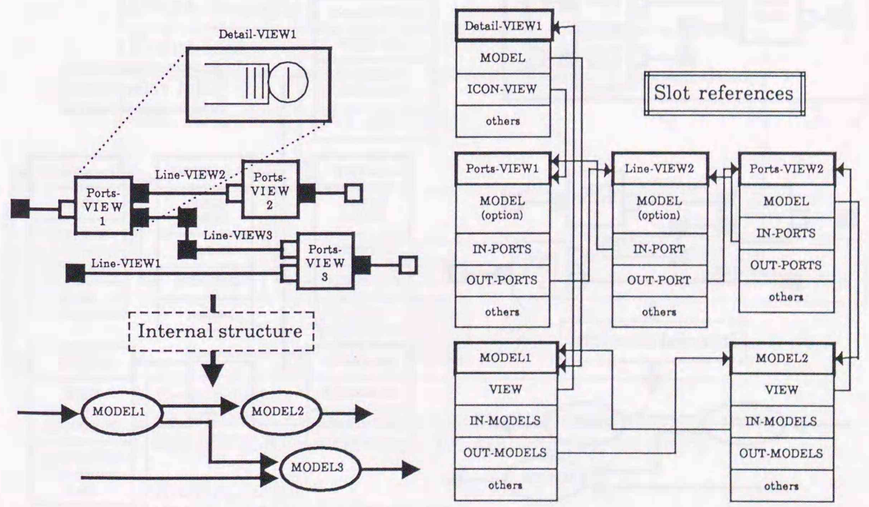


図2.10　Detail-VIEWおよびPorts-VIEWとLine-VIEWの接続関係とそれらの内部構造とスロット参照関係

MODELの内部で定義される。図2.10の左図に示されるように、Detail-VIEWをアイコン化するとPorts-VIEWとなり、このPorts-VIEWの入出力が、Detail-VIEWに割当てられているMODELの入出力となる。図2.10の右図に示されるように、Detail-VIEWのスロットICON-VIEWには、アイコンであるPorts-VIEWが格納される。

(3) Hierarchy-VIEW：対象モデルを階層的に記述するためのVIEWであり、この内部で対象モデルの部分系を記述できる。図2.11の左図に示されるように、Hierarchy-VIEWもアイコン化するとPorts-VIEWとなり、一つの構成要素として用いることができる。図2.11の右図にスロットの参照関係が示されるように、Hierarchy-VIEWのアイコンであるPorts-VIEWの入出力は、Hierarchy-VIEWの内部で記述されている部分系の入出力となる。多くの構成要素からなる対象モデルを階層的に記述できる。

図2.12の上部に示されるようにPort-VIEWとLine-VIEWおよびHierarchy-VIEWを用いてダイアグラムを階層的に構成した場合、図2.12の下部に破線で示されるように各VIEWに割当てられているMODELの入出力関係が定義される。このようにしてダイアグラムを画面上で作図することにより、対象システムの構成の定義を視覚的に行うことができる。異なる構成をもつ種々の対象システムについて数回のシミュレーションを行う場合にも、その構成の定義に要する時間と作業量は少ない。

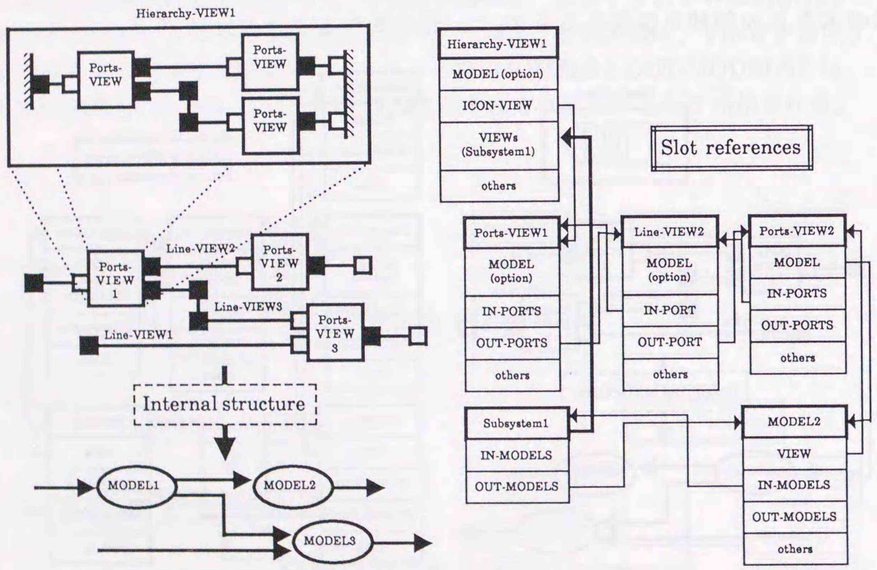


図2.11 Hierarchy-VIEWおよびPorts-VIEWとLine-VIEWの接続関係とそれらの内部構造とスロット参照関係

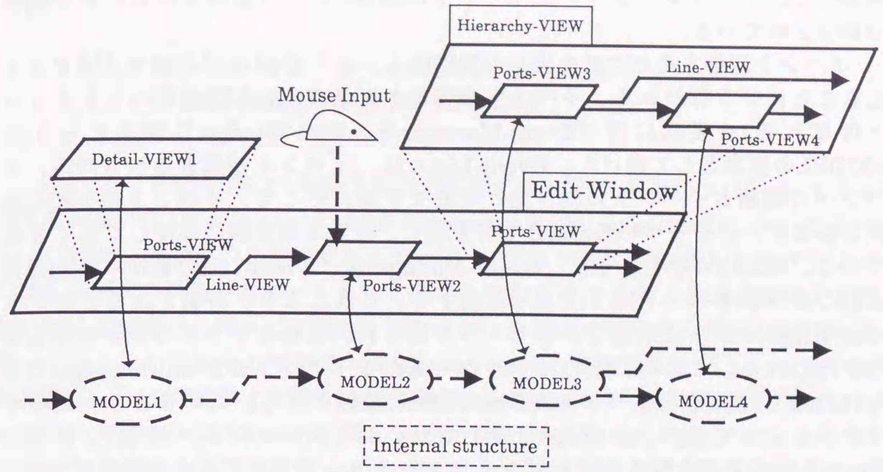


図 2.12 Detail-VIEW, Ports-VIEW,Hierarhy-VIEWとそれらのMODEL間の関係

2.2.3 シミュレーションの動作方式

ディスク･サブシステムなどの計算機システムをシミュレーション対象とする場合、その動作はイベント駆動方式[71]によりシミュレーション可能である。

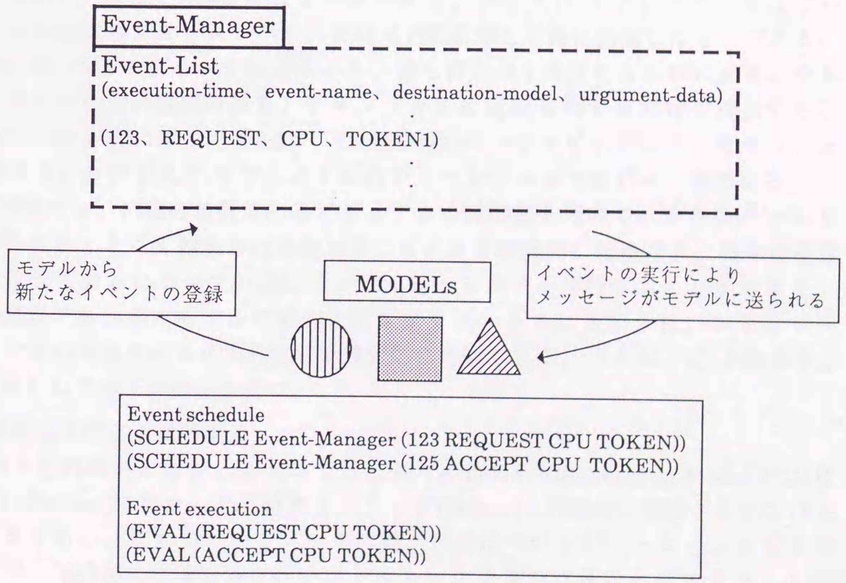


図2.13 イベント駆動による動作

FESには、シミュレーションをイベント駆動方式により動作させるための機能が用意されている。

イベント駆動方式では、モデルの動作はメッセージ·パッシングでイベントを送ることにより行われる。イベントを管理するためEvent-List(イベントキューと呼ばれる)を要素に持つEvent-Managerを、Edit-Windowに割当てられるMODELの要素として設けた。Event-Listとは、イベントが起きるべき時間、イベントの種類とイベントの送り先となるモデル、そして、イベントの生起に必要となるデータを一つのEventとして記述し、それを時間順に保持しておくものである。図2.13に示すように、Event-Managerは、Event-Listの最初の要素(最初に起こるべきイベント)を引き出し評価する。これにより、評価されるイベントの送り先のモデルに対してメッセージが送られ、このモデルの状態が変化し動作が行われる。この時、そのモデルから新たなイベントがEvent-Managerに送られたならば、そのイベントがEvent-Listに登録される。このような一連のサイクルによってイベントが次々と実行され、シミュレーションが実行される。Event-Listの内容を変えなければシミュレーションの途中で中止や再開が自由にでき、これにより対話的操作が可能となる。

2.3節　FESの応用シミュレータ·システム例
待ち行列網シミュレータ: CAB

本説では、FESを用いて開発された対話型グラフィカル·シミュレータである待ち行列網[70][71]シミュレータ: CAB (Computer Architect's Board)について述べる。CABは、本研究の主目的であるディスク·サブシステムの高スループット化に関する研究の性能評価ツールとして開発されたものである。FESが提供する機能の他にCAB独自の種々の特徴的な機能を持つ。FESの提供する機能がCABの開発でどのように用いられているかを述べる。また、実際のシミュレーションの例を挙げ、CABが提供する種々の動作機構をもつ基本モデルが計算機システムのモデリングにおいてどのように用いられているかを述べる。実際のディスク·サブシステムのシミュレーションにおいてこれら種々の動作機構をもつ基本モデルがどのように用いられているかは、3章の性能評価のところで説明を加える。さらに本節では、FESを用いて開発されたその他のシミュレータであるペトリネット[72][73]シミュレータ: PABと論理回路シミュレータ: LABについても簡単に説明し、FESの有用性を示す。

2.3.1　CABの特徴

CABは、待ち行列網シミュレータであり、ディスク·サブシステムをはじめとする計算機のシステムレベルにおける性能評価を目的に開発したものである。FESの持つMVモデリング機能により、待ち行列網を記述するために必要となる基本モデルの動作機構のみを、テキスチャルに定義·登録するだけで開発することができた。先に示した図2.1が、CABの画面ハードコピーであり、キャッシュメモリ付きのディスク·サブシステムのシミュレーションを行った例である。

CABでは、FESの特徴であるビジュアル合成機能を用いて、テキスチャルな記述をすることなく対象となる計算機システムを視覚的に記述でき、種々の構成の対象システムについて対話的にシミュレーションを実行することができる。構成要素である基本モデルの動作状況もグラフィカルに表示され、シミュレーション中の対象モデルの動作過程を視覚的に理解できる。さらに、CAB独自の拡張機能として以下の特徴を持つ。

(1) 状態遷移による動作記述

待ち行列網シミュレーションでは、個々の待ち行列(Queueing-Model)における窓口(Server)でのサービス時間と、ジョブ(Token)の到着時間間隔が重要なパラメータである。これらのパラメータの種々の値についてシミュレーションすることで、網全体の性能評価を行いボトルネックを見つけることができる。しかし、窓口でのサービスの内容については考慮されないため、サービスの内容を

細かく記述できない。そこで、CABでは、Serverを確率遷移オートマトンとしても記述できるようにした。Serverに内部状態を持たせQueueing-Modelへの入力であるTokenにラベルを与え、図2.14に示されるように、このラベルの種類によってServerでの処理時間を変えたり、出力するTokenのラベルの種類を決めることができるようにした。図の例では、初期状態Q0のServerにラベルnをもつTokenが到着した場合、確率1で(必ず)200時間サービスをしてラベルmをもつTokenを出力し、サービス後の状態はQ2となる。また、初期状態Q0のServerにラベルmをもつTokenが到着した場合には、確率0.5で200時間サービスをしてラベルlをもつTokenを出力し状態Q1となる、あるいは、確率0.5で100時間サービスをしてラベルnをもつTokenを出力し状態Q1となる。このように、Queueing-ModelのServerを確率遷移オートマトンとしても記述できるようにしたことにより、通常のQueueing-Modelでは表現できない細かな動作のモデリングも可能となった。

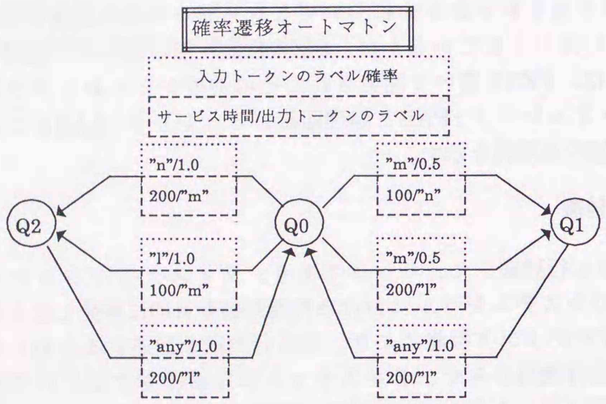


図2.14 サーバ動作の状態遷移による表現

(2) 対象システム記述における簡便性

CABでは、シミュレーションの対象となる計算機システムの動作を簡便に記述できるよう、後述する種々の基本モデルをあらかじめ用意している。対象となる計算機システムは、待ち行列網モデルで表される。そこで、Tokenのフローに関して種々の形態の制御が行えるように、いくつかのフロー制御用の基本モデルを用意した。例えば、ペトリネットにおけるプレースとトランジションと同様の機能を持つ基本モデルを用意した。これにより、要求駆動型の処理(Token)の流れも実現でき、計算機システムの動作記述における柔軟性が増した。

2.3.2 CABの提供する待ち行列網の記述用モデル

待ち行列網シミュレーションにより計算機システムの性能評価を行うために、あらかじめ定義·登録されている基本モデルには以下に挙げるものがある。(図2.15参照)

(1) TOKEN : ジョブ(処理)を表すモデルである。処理の種類を示すラベルと優先順位を示すプライオリティ·ナンバを要素として持つ。また、フロー制御のための種々の要素(アドレス、パス番号など)を持つ。図2.15に示すようなグラフィックス·イメージで表示される。

(2) CONNECTOR : 多入力、多出力であり、TOKENのフロー制御のためのモデルである。乱数方式や、ラウンドロビン方式により、出力先を決めることができるだけでなく、TOKENの持つ要素(ラベル、アドレス、パス番号など)の値に従って、出力先を決めることができる。

(3) RESERVER & RELEASER : リソース使用の排他制御をするためのモデルである。TOKENがRESERVERを通過すると、このRESERVERに対応するRELEASERをそのTOKENが通過するまで、そのRESERVERはロックされ、別のTOKENを通過させることができない。

(4) PLACE & TRANSITION : これもフロー制御のためのモデルであり、同期および排他制御の記述に用いられる。PLACEは、待ち行列あるいはバッファとして機能する。その動作は、ペトリネットにおけるプレースと同様であり、排他制御に用いることができる。PLACEの内部には、待ち行列(バッファ)の長さがグラフィカルに表示される。TRANSITIONも、ペトリネットにおけるトランジションと同様の動作をし、同期制御に用いることができる。

(5) CHANGER : TOKENが持つ種々の要素(ラベル、アドレス、パス番号など)の値を強制的に変更するためのモデルである。

(6) QUEUEING-MODEL : サービスをする窓口(SERVER)と待ち行列(QUEUE)を表すモデルである。前述したように、TOKENを入出力とした確率遷移オートマトンとしてサービスすることもできる。

(7) TOKEN-PRODUCER : TOKENを確率分布で表されるある時間間隔で生成し、システムに送りだすモデルである。あらかじめ用意されている確率分布には、指数分布、アーラン分布、一様分布がある。時間間隔のデータをリスト形式で与えることにより、それに従ってTOKENの生成を行うこともできる。

(8) TOKEN-COUNTER : システムで処理を終えたTOKENの回収を行ったり、TOKENの通過量を計りシミュレーションの終了制御を行うモデルである。

以上の基本モデルのうち、(2)(3)(4)(5)は、Ports-VIEWに割当てられるMODELとして、その動作機構のみがテキスチャルに定義されている。図2.15に示されるように、モデルの種類によってPorts-VIEWのフォームを変えており、

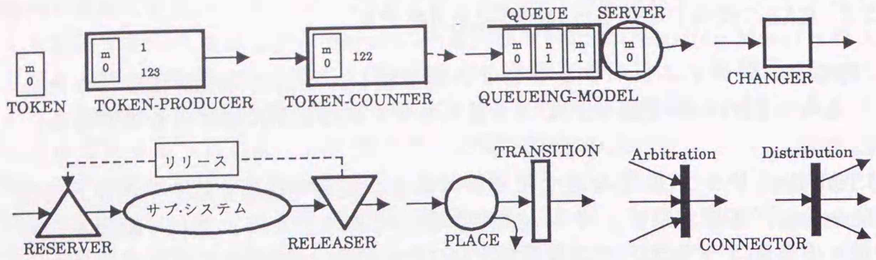


図 2.15　待ち行列網システムの記述用モデル

Ports-VIEWに割当てられているMODELが何であるかをディスプレイ画面上で識別できる。

また、(6)(7)(8)は、Detail-VIEWに割当てられるMODELとして定義されており、図2.15に示されるようなグラフィックス・イメージで表示される。QUEUEING-MODELでは、QUEUEとSERVERの内部にTOKENのグラフィックス・イメージが表示され、待ち行列の長さやSERVERの状態が視覚的に把握できる。TOKEN-PRODUCERでは、システムに送り出したTOKENの種類と数がグラフィカルに表示される。TOKEN-COUNTERでは、通過したTOKENの数がグラフィカルに表示される。シミュレーション中には、これらのグラフィックイメージがリアルタイムで変化し、対象システムの動作状況をアニメーションを見るごとく視覚的に把握できる。

上述したこれらのモデルを用いることで、TOKENのフロー制御を柔軟に表現でき、種々の機構を持つ対象システムを記述できる。MVモデリングにより、対象となる計算機システムの構成要素の動作機構のみをプログラムにより定義し、新たなMODELとして登録することができ、その固有の動作を細かく記述することも可能である。また、Hierarchy-VIEWを用いて、対象システムの部分シ

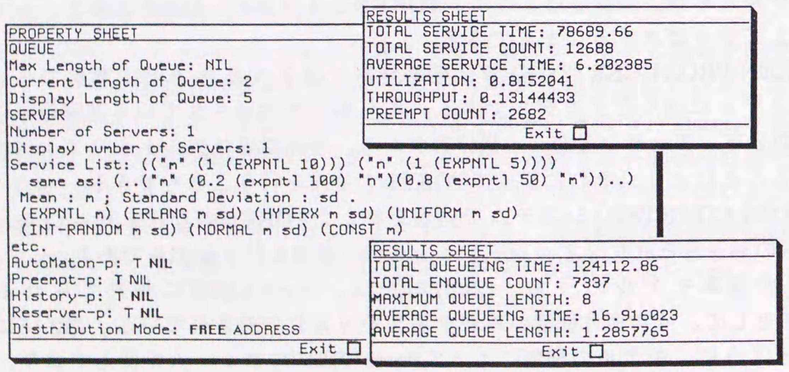


図 2.16　QUEUEING-MODELのメニューシート

ステムをこのVIEWの内部で記述することで、階層的に構成の定義を行うことができる。

図2.15に示した各モデルは、それぞれプロパティおよび結果を保持している。図2.16に示すように、プロパティの設定や結果の表示は、専用のウィンドウ(プロパティシートおよびリザルトシート)を開いて、メニューを選択することにより対話的に行うことができる。

2.3.3　CABのシミュレーション例

図2.17に示されるのは、文献[70]に挙げられている待ち行列網の例である。これをCentral Server Queueing Networkと呼ぶ。これは、CABの開発段階でシミュレータの動作検証のための評価例として用いた待ち行列網である。以下では、これを例にCABによるシミュレーション例を述べる。文献[70]は、待ち行列網モデルに基づいた計算機システムの性能評価方法について述べている。この文献には、C言語とSMPLという(Queueing-Model記述用の)ライブラリィを用いて計算機システムの動作を記述し、シミュレーション実験した結果がいくつか記載されている。

図2.17に示されるシステムは、1つのCPUと4つのディスク装置で構成される計算機システムを表している。これらは、それぞれQueueing-Modelで表されている。このシステムには、9つの端末装置が接続されており、各端末装置から出された仕事はすべて、CPU、ディスク装置と続けて処理されると仮定する。適当な処理時間で処理され、その後、再び端末装置から仕事が出され、これが繰り返されると仮定する。9つの端末装置の中、3つの端末装置から出される仕事は、残り6つの端末装置から出される仕事に優先して処理されるとする。すなわち、2種類の仕事(ジョブ)n0およびn1がそれぞれ6つおよび3つあり、CPUではn1はn0に優先して処理される。CPUでの処理時間は、n0が10msec、n1が5msecの平均値を持つ指数分布である。ディスク装置での処理時間は、n0、n1ともに30msecの平均値、その1/4の7.5msecの標準偏差を持つアーラン分布である。CPUからディスク装置へのアクセスは、4つのディスク装置の中の1つへランダムに行わ

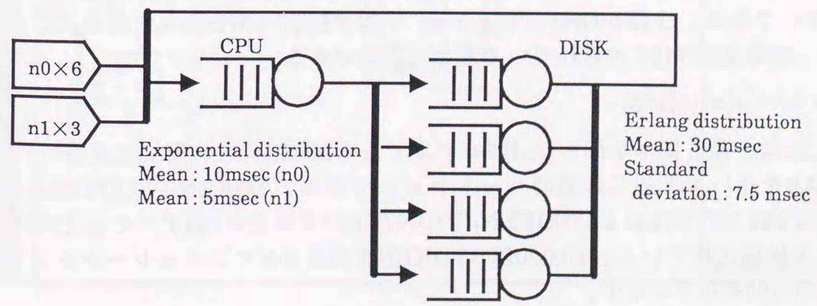


図2.17　Central Server Queueing Networkシステムの構成

れる。すべての端末装置から出された仕事の合計が10,000になった時点で、シミュレーションを終了させる。シミュレーション結果として、各Queueing-Model毎に、平均処理時間、利用率、スループット、平均待ち時間、平均の待ち行列長などが得られる。これらの結果を評価して、システム全体の性能評価を行う。

図2.18は、この計算機システムをCABを用いて記述したEdit-Windowの画面ハードコピーである。Edit-Window内部には、VIEWの表示形態と構成の仕方を変えた3つの同一システムが表示されている。5つのQUEUEING-MODEL、2つのTOKEN-PRODUCER、1つのTOKEN-COUNTERと、3つのCONNECTORでそれぞれ構成されている。2つのTOKEN-PRODUCERは、それぞれn0,n1にあたるTOKENを6個、3個生成しシステムへ送りだす。9つのTOKENは、まずCPUに対応するQUEUEING-MODELで処理される。処理が終えたTOKENはCONNECTORへ送られる。CONNECTORは、ディスク装置に対応する4つのQUEUEING-MODELの中の1つをランダムに決め、それにTOKENを送る。処理が完了したTOKENは、TOKEN-COUNTERを通過し、再びCPUに対応するQUEUEING-MODELへ送られる。TOKEN-COUNTERは、通過したTOKEN数を計測する。それが10,000になった時点で、Event-Managerに対してシミュレーション実行の中断メッセージを送る。このような流れによりシミュレーションが実行される。

図2.18の右図は、図2.18の左図に示されたTOKEN-PRODUCER,TOKEN-COUNTER,QUEUEING-MODELの各Detail-VIEWを開き、状態をグラフィカルに表示したものである。シミュレーション中には、これらの表示がリアルタイムで変化し、各構成要素の状態が視覚的に把握できる。図2.18の下図は、図2.18の上図に示された構成をHierarchy-VIEWを用いて階層的に定義したものである。

各QUEUEING-MODELは、平均の待ち行列長、平均待ち時間、平均サービス時間などの値を保持している。図2.19は、CPUとディスク装置について平均待ち行列長のグラフを表示させたGraph-Windowと、すべてのQUEUEING-MODELについて、それらが保持する数値結果を表示させたReport-Windowの画面ハード･コピーである。上図がGraph-Window、下図がReport-Windowである。この結果は、参考文献[71]に示されているものと同値である。

[シミュレーション性能]

図2.20は、最も簡単な待ち行列網モデルであるM/M/1待ち行列モデル[70][71][72]を、CABを用いて記述した画面ハードコピーである。TOKEN-PRODUCERとQUEUEING-MODELおよびTOKEN-COUNTERがそれぞれ1個ずつの合計3個のモデルで構成されている。10,000個のTOKENを発生させてシミュレーションした場合の時間を以下に示す。

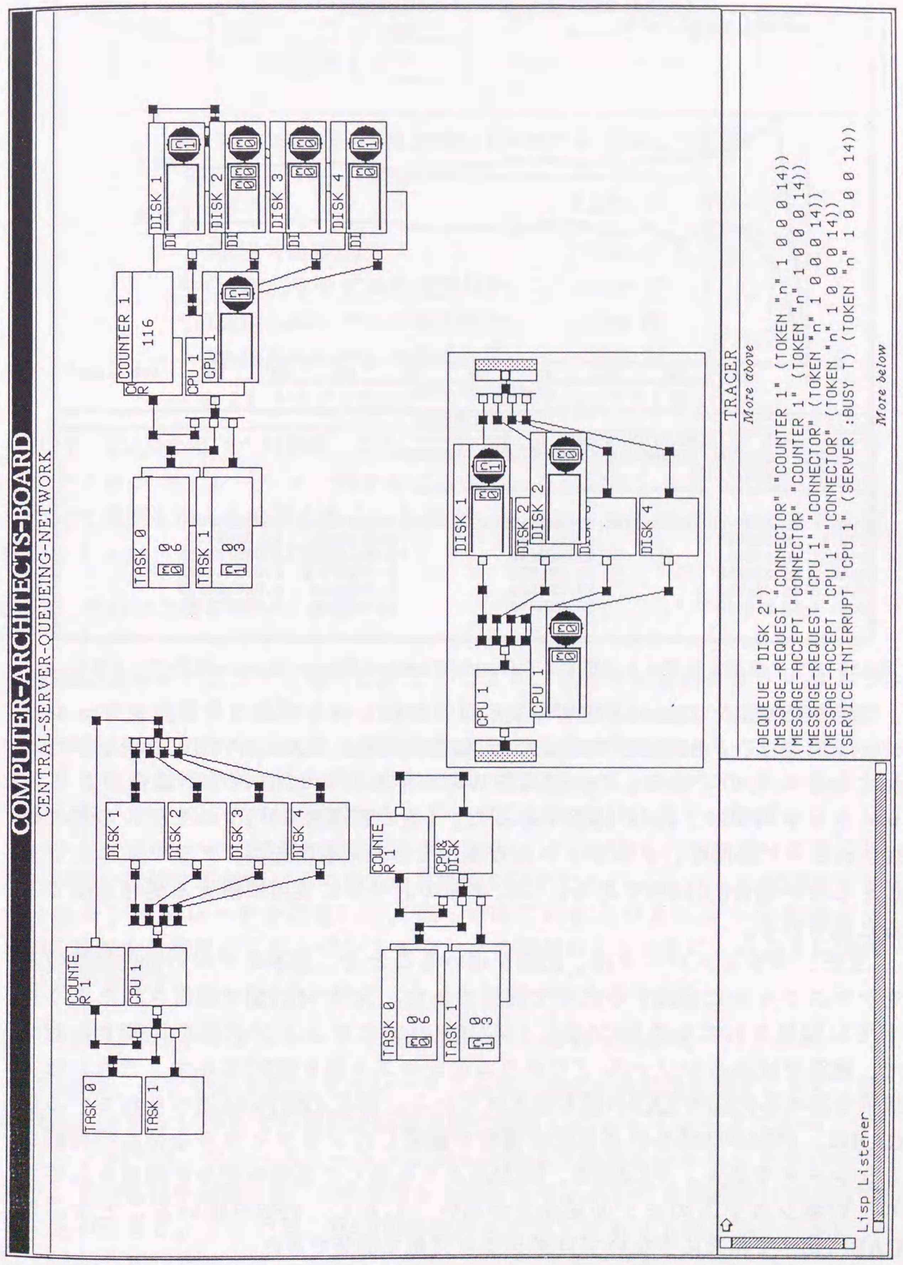


図2.18 Central Server Queueing Networkを定義したCABのEdit-Window画面ハードコピー

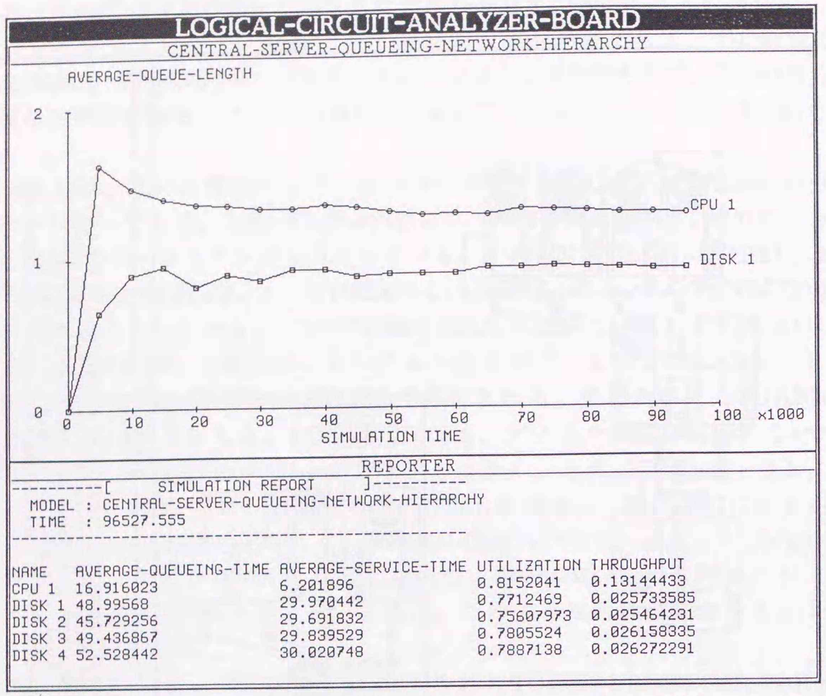


図2.19 グラフ表示とリポート表示をしたGraph-WindowとReport-Window画面ハードコピー

図2.20左図は、Detail-VIEWの表示が行われているものである。シミュレーション時間は、約8分26秒であった。図2.20右図は、Detail-VIEWの表示が行われていないものである。Detail-VIEWの表示が行われていない場合のシミュレーション時間は、約4分15秒であった。1個のTOKENが3つのモデルの間で処理される平均時間は、グラフィカルな表示をした場合0.05秒、グラフィカルな表示をしない場合0.025秒である。したがって、十分に実用に耐える処理速度であると思われる。

また、本シミュレータは、FESを用いることで、基本モデルの動作機構のみをテキスチャルに定義するだけで開発できた。文献[74][75]で報告されたようにすでに開発されていた旧CABと、本CABのプログラミング量を比較する意味で、概算ではあるがソース・プログラムのテキスト量を表2.1に示す。これには、以下で述べるPABやLABの値も含まれている。参考文献[75]に述べられている旧CABは、FESが開発される以前に著者が開発したグラフィカルな待ち行列網シミュレータである。本CABは、旧CABよりも多くの基本モデルが用意されており、対象システムのモデル記述力が高い。しかし、FESを用いることで、旧CABに比べて格段に少ないプログラミング量で開発できた。

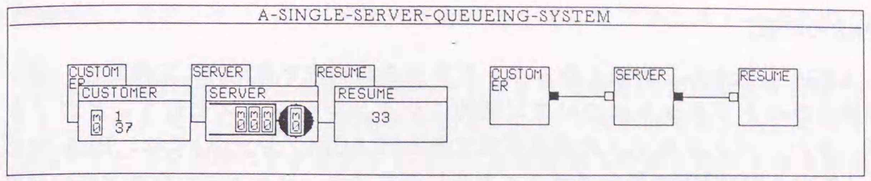


図 2.20 CABを用いてM/M/1待ち行列モデルを記述した例

・旧CAB	5,500 行
・FES(共通部分)	7,500 行
・CAB(基本モデルの定義部分)	3,400 行
・LAB(基本モデルの定義部分)	1,300 行
・PAB(基本モデルの定義部分)	700 行

表 2.1 システム毎のソースプログラムのテキスト量

以上、CABを用いた計算機システムの性能評価の例を示した。ディスク・サブシステムのシミュレーション例を挙げる前に、その対象システムの構成と動作について説明を加える必要がある。したがって、ディスク・サブシステムの具体的なシミュレーション例は3章において詳しく述べる。

2.3.4 その他のシミュレータ例

FESはMVモデリングを導入したことにより、種々の対話型グラフィカルシミュレータを開発する汎用のツールキット・システムとなった。先のFESの特徴で述べたように、ダイアグラム表現によりその構成が表され、イベント駆動方式によりその動作が表される事物であれば、FESのシミュレーション対象として扱える。すでに、待ち行列網シミュレータCABの他に、以下に挙げるペトリネットシミュレータ:PABと論理回路シミュレータ:LABをFESを用いて開発した。FESの提供する機能やその特徴の有効性を示すために、これらの対話型グラフィカルシミュレータを開発した。以下ではこれらのシミュレータを挙げ、FESの提供する機能がこれらのシミュレータの開発および特徴にどのように現れているかを示す。

(1) ペトリネットシミュレータ:PAB(Petri-net Analyzer Board)

PABは、ペトリネット・シミュレータであり、FESの有用性を示すと共に、計算機のシステムレベルでの性能評価を目的に開発したシミュレータである。ペトリネット[73]は、プレースとトランジションにより構成されるネットモデルであり、並列処理や同期制御、排他制御を含むシステムの動作を抽象化して表現することができる。ここでは、PABの特徴およびペトリネット記述用モデルについて解説するのみで、ペトリネット理論についての説明は他の文献に譲る。

[PABの特徴]

FESの特徴から、対象となるペトリネットの構成を視覚的に定義でき、種々の構成のペトリネットについて対話的にシミュレーションを行うことができる。また、ペトリネットの構成要素であるPLACE(プレース)では、PLACEに入っているTOKENの数をグラフィカルに表示でき、シミュレーション中の対象系の状態や動作を視覚的に把握できる。PABでは、Standard Petri-Net[72]、およびTimed Petri-Net[72]をシミュレーションすることができる。Timed Petri-Netでは、時間要素としてTRANSITION(トランジション)にサービス時間、PLACEに待ち時間を持たせており、QUEUEING-MODELにおけるQUEUEとSERVERと同様の機能を実現している。実行結果は、メニューを開いて選択することで、グラフおよび数値で表示することができる。

[ペトリネットの記述用モデル]

(1) PLACE : ペトリネットにおいてプレースにあたるモデルである。TOKENの平均の待ち時間、待ち行列の平均の長さや最大の長さなどを結果として保持している。

(2) TRANSITION : ペトリネットにおいてトランジションにあたるモデルである。サービス時間(確率分布)を要素として持ち、平均のサービス時間や利用率などを結果として保持している。

各モデルはプロパティを保持し、メニュー選択により対応するウィンドウを開いてこれを表示したり変更を行うことができる。ここで定義されているPLACE、TRANSITIONは、CABで定義されているものとは機能やメッセージに互換性がなく、別のモデルとして定義･登録されている。

[シミュレーション例]

図2.21は、文献[72]においてペトリネットでモデリングされている計算機システムの構成である。n台のプロセッサ(P)とローカル･メモリ(LM)、および共有メモリ(CM)とで構成される。各プロセッサとローカル･メモリはローカル･バスで接続され、共有メモリへは、グローバル･バスを通してアクセスすることができる。

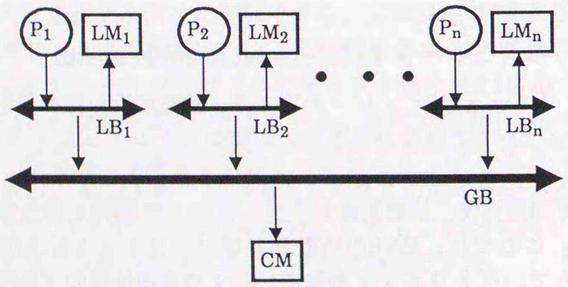


図2.21 マルチプロセッサ･システムの構成

図2.22はPABのEdit-Windowの画面ハードコピーであり、プロセッサ台数を8台としてこのシステムを記述したものである。このシステムでは、各プロセッサが共有メモリを同時にアクセスする場合に起きるバスのボトルネックを評価する。図2.21において、プロセッサとローカル･メモリの処理は共有メモリをアクセスしないローカルな処理であるから、1つの処理要素としてまとめることができる。図2.22において、'PE'とラベルの表示されているモデルはトランジションであり、バスをアクセスしないローカルな処理を表す。'BUS'とラベルの

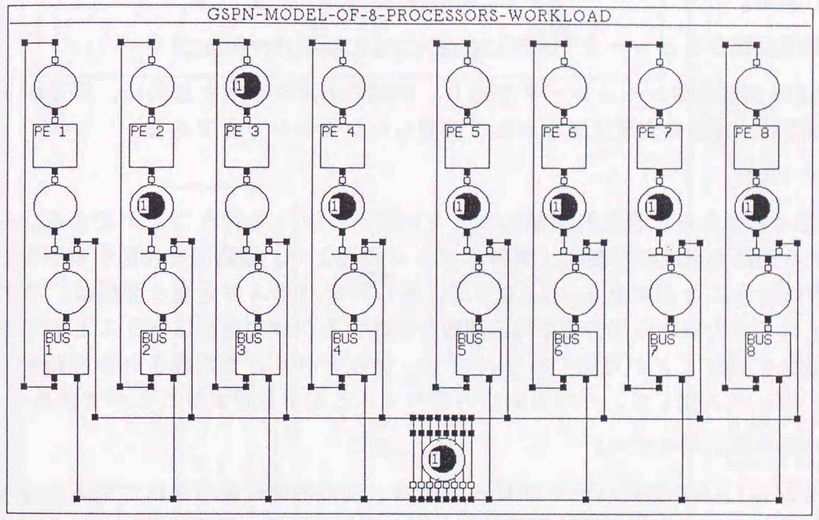


図2.22 マルチプロセッサ･システムを定義したPABのEdit-Window画面ハードコピー

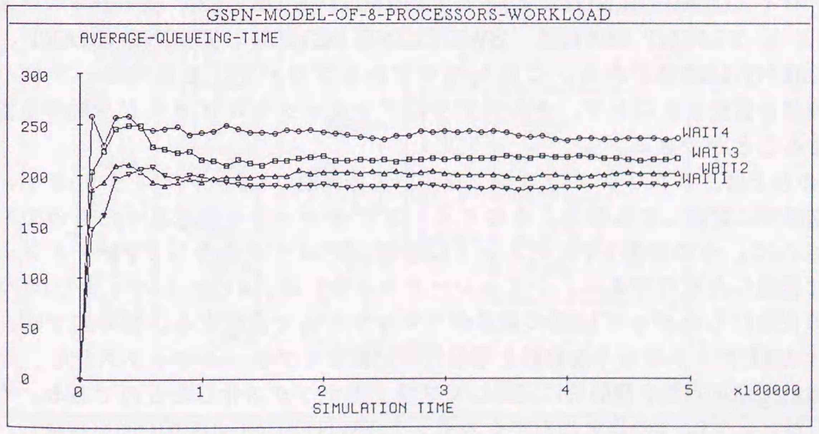


図2.23 グラフ表示をしたGraph-Windowの画面ハードコピー

表示されているモデルもトランジションであり、バスをアクセス中の処理を表す。中央にあるのがプレースであり、バスの使用条件を与える。このプレースにトークンが存在していれば、バスが使用可能である。ローカルな処理時間を平均が30(msec)の指数分布、バスをアクセスする場合の処理時間を平均が70(msec)の指数分布とし、シミュレーション時間を500,000(msec)としてシミュレーションを行った。シミュレーション中、各PLACEが保持しているTOKEN(黒丸)とその数がグラフィカルに表示される。リアルタイムでそれが変化し、この対象システムの動作状況が視覚的に把握できる。図2.23は、8台の中の4台のプロセッサについて、バスの平均待ち時間をグラフ表示したGraph-Windowの画面ハードコピーである。

(2)　論理回路シミュレータ：LAB(Logical-circuit Analyzer Board)

LABは論理回路シミュレータであり、FESの有用性を示すと共に、ディジタル回路設計、評価の支援ツールとして開発したシミュレータである。

[LABの特徴]

FESの特徴から、視覚的･対話的にシミュレーションを行うことができる。対象とする回路の構成の定義は、ディスプレイ画面上で、回路図を作図する要領で視覚的に行うことができる。LABでは、新しいデバイスの定義と登録は、そのデバイスの入力値から出力値を得る関数を定義するだけで容易に行うことができる。階層モデリングが可能であり、いくつかのデバイスで構成される回路をブラックボックス化して、一つの新しいデバイスとして使用することができる。

[論理回路の記述用モデル]

図2.24はLABの画面ハードコピーである。図の右側に表示されている基本モデルを用いて、いくつか回路を構成した例を示している。論理回路における基本的なデバイス(AND,OR,NOTなど)としてPRIMITIVE-DEVICE、表示器および入力器としてLIGHT-DEVICE、SWITCHING-DEVICE、TIMING-CHART、7SEGMENT-LED等がある。これらのモデルもプロパティを保持し、プロパティの値を変更することで、そのモデルのグラフィックス･イメージや動作を変化させることができる。

図の左上は、T-フリップ･フロップを用いて非同期の4ビット･バイナリ･カウンタを階層的に定義したものと、そのタイミング･チャートを表示させたものである。さらに、その出力を7セグメントLED用のデコーダを介して7セグメントLEDに接続したものである。シミュレーション中には、4ビット･バイナリ･カウンタの出力にしたがってLEDの表示がリアルタイムで変化する。図の左下は、XORとANDゲートから半加算器を階層的に定義してブラックボックス化し、それを用いて全加算器を階層的に定義してブラックボックス化したものである。その入力器として0、1の値を出力するスイッチ(SWITCHING-DEVICE)を接続し、

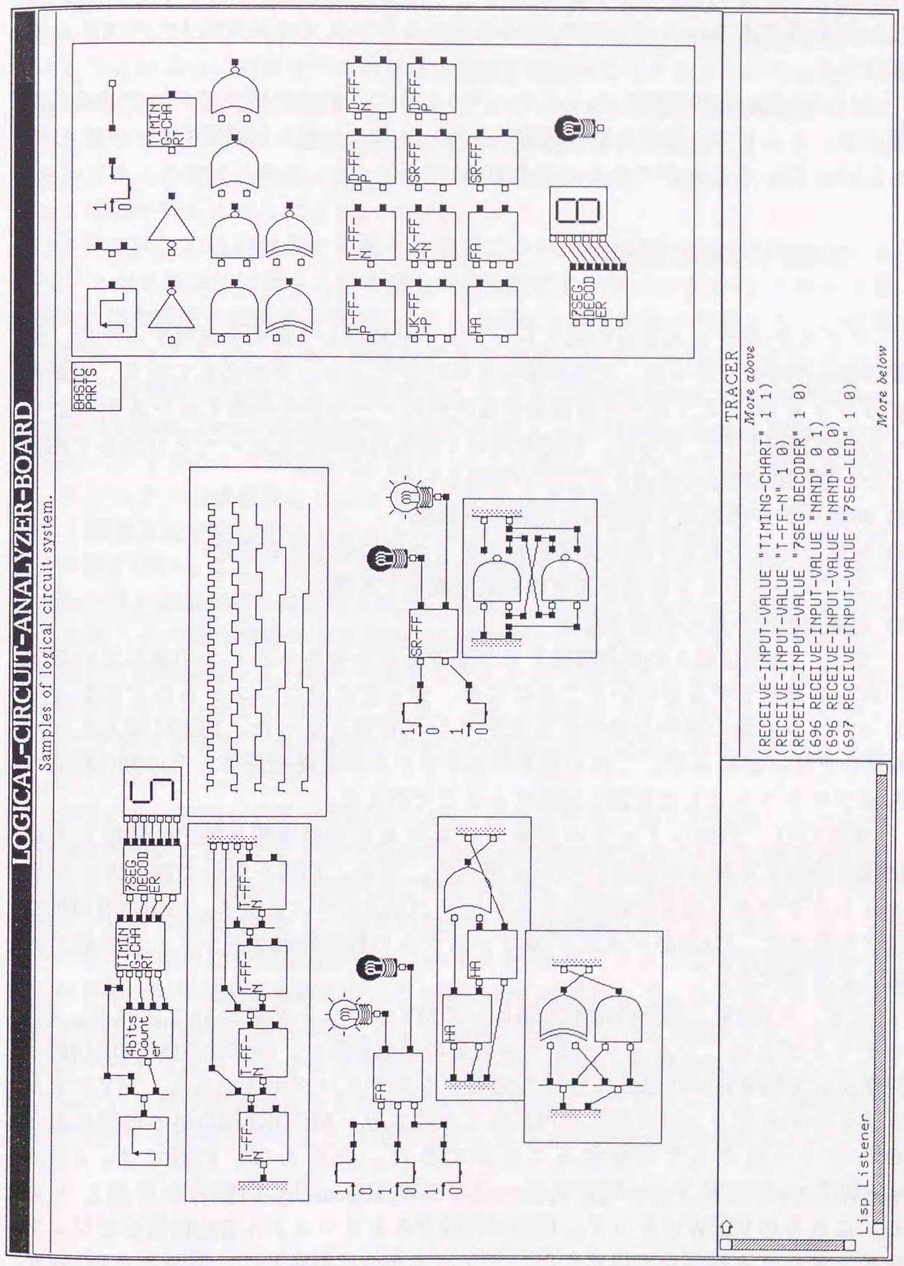


図2.24　LABの画面ハード･コピー

出力器としてランプ(LIGHT-DEVICE)を接続している。図の中下は、NANDゲートからSR-フリップ･フロップを階層的に定義してブラックボックス化したものである。スイッチのオン/オフによりランプの表示がリアルタイムで変化する。

M.G.Walkerらの開発したシステム[4]のように、FESの持つビジュアル合成機能を用いることで、既存の論理回路シミュレータに対する回路図エディタとしてもLABを用いることができると思われる。

2.4節　2章の要約

ディスク･サブシステムの高スループット化に関して研究を遂行する上で、性能評価は不可欠であった。性能評価のためにシミュレータが必要であり、対話型グラフィカル･シミュレータを開発するためのツール群を提供するシステムとしてFESを開発した。さらに、FESを用いて待ち行列網シミュレータ：CABを開発した。CABはFESの提供する以下の機能をもつ。

1) 対象モデルの構成を視覚的に定義する機能
2) 対象モデルの状態や動作状況をグラフィカルに表示する機能
3) シミュレーション結果を種々の形式で表示する機能
4) 対話型操作のための統合環境

また、CABは種々の動作機構をもつ基本モデルをあらかじめ用意している。これらの基本モデルを用いることにより、ディスク･サブシステムなどのシミュレーション対象の詳細な動作をもモデリング可能となった。FESに導入されたMVモデリングにより、これら種々の基本モデルの定義･登録は、その動作機構のみをプログラムとして記述し定義することで行えた。

本章では、FESシミュレーションシステムの設計思想とFESの特徴およびFESが提供する種々の機能について述べた。また、FESを用いて開発されたシミュレータである待ち行列網シミュレータ：CABについて述べ、FESの有用性を示すとともに、CABがディスク･サブシステムの性能評価ツールとして有益であることを述べた。

まず、2.1節で、FESの設計思想としてMVモデリングについて述べた。MVモデリングとは、シミュレーション対象の構成要素を、動作記述(MODEL)部と視覚表示(VIEW)部の2部分に分けて定義するモデリング手法である。MVモデリングと呼ぶモデリング手法を採用したことにより、MODEL部以外の機能をあらかじめツールとして用意することができた。たとえば、FESでは、4種のVIEW(Line-VIEW, Ports-VIEW, Detail-VIEW, Hierarchy-VIEW)を用意している。これらのVIEWによって、FESの特徴であるビジュアル合成機能やビジュアルインスペクタ機能が実現できた。ビジュアル合成機能とは、画面上でVIEWを組合せ作図することにより、対象モデルの構成の定義が行える機能である。ま

た、Detail-VIEWを用いることにより、このDetail-VIEWに割当てられているMODELの状態や動作状況をグラフィクス･イメージで視覚表示することができた。これがビジュアルインスペクタ機能である。さらに、Hierarchy-VIEWを用いることにより、対象モデルの構成の定義を階層的に行えるようになった。これらのVIEWを用意したことにより、シミュレーション対象の構成の定義が容易に行えるようになった。異なる構成の種々の対象システムに関しても、それら対象システムの構成の定義に要する時間が非常に短くなり、効率的にシミュレーション実験が行えるようになる。

2.2節では、FESが提供する種々の機能について述べた。FESでは、シミュレーションを対話的に行うための統合環境を提供するウィンドウ･システムと視覚的な入出力機能を提供するMVシステムで構成されている。対象システムの構成の定義、シミュレーション結果のグラフ表示、レポート作成、イベントトレースの各機能に対応して、4種のウィンドウ(Edit-Window, Graph-Window, Report-Window, Trace-Window)をシステム･ウィンドウ(Simulation-Window)のサブ･ウィンドウとして用意した。MVシステムでは、MVモデリング機構を提供し、ビジュアル合成機能とビジュアルインスペクタ機能を実現した。また、イベント駆動方式で動作するシミュレーション･ドライバを用意した。これらのツール群を用いることにより、対話型で視覚的な入出力機能をもつ待ち行列網シミュレータ：CABが簡単に開発された。さらに、FESに導入されたMVモデリングにより、その動作機構であるMODEL部のみをテキスチャルに記述することで、異なる動作機構をもつ種々の基本モデルを追加･登録できる。CABは種々の基本モデルをあらかじめ用意しており、それら基本モデルを用いることでディスク･サブシステムなどの計算機システムの詳細な動作記述が行えるようになった。

2.3節では、FESの応用シミュレータ･システムである待ち行列網シミュレータ：CABについて述べた。FESの提供する種々の機能がCABのもつ特徴にどのように現れているかを述べ、FESの有用性を示した。また、CAB独自の特徴について述べ、異なる動作機構をもつ種々の基本モデルを用意したことにより、シミュレーション対象の動作記述力が高く、ディスク･サブシステムなどの計算機システムの詳細な動作記述をも行えることを述べた。実際にシミュレーションを行った例を挙げ、CABの有用性を示した。さらに、FESを用いて開発されたその他のシミュレータであるペトリネット･シミュレータ：PABと論理回路シミュレータ：LABについて簡単に説明を加えた。これらの開発およびこれらのもつ特徴に、FESが提供する機能がどのように現れているかを述べFESの有用性を示した。

第3章 マルチポート·ページメモリをディスクキャッシュとして用いたディスク·サブシステムDIMPのアーキテクチャ

本章以下では、ディスク·サブシステムの高スループット化について述べる。まず、本章ではマルチポート·ページメモリ(MPPM : Multi-Port Page Memory)をディスクキャッシュとして階層的に用いたディスク·サブシステム：DIMP(DIsk subsystem with MPPM)の構成とキャッシュ動作について述べる。また、シミュレーションにより得られた処理性能を挙げ、DIMPは高スループットが得られるディスク·サブシステムであることを示す。

以下では、3.1節で、まず従来のキャッシュメモリ付きディスク·サブシステムのボトルネックについて解説する。3.2節で、DIMPのアーキテクチャについて解説する。MPPMの構成と動作について述べ、その後、DIMPの基本的な構成例を示す。3.3節ではキャッシュ動作を示す。ここでは、キャッシュデータのコヒレンシを考慮しない場合のキャッシュ動作について述べる。キャッシュデータのコヒレンシを考慮した場合のキャッシュ動作および処理性能については4章で述べる。3.4節では、シミュレーションにより得られた従来型ディスク·サブシステムの処理性能を示しボトルネックを指摘する。また、DIMPの処理性能を示し、従来のディスク·サブシステムの処理性能と比較しDIMPの有効性を述べる。3.5節で本章のまとめを述べる。

3.1節 従来のディスク·サブシステムのボトルネック

本節では、従来のキャッシュメモリ付きディスク·サブシステムの構成図と基本的なキャッシュ動作図を示し、ディスク装置とディスクキャッシュ間のパス(以下D-Cパスと呼ぶことにする)がボトルネックであることを指摘する。以下、パスとはデータ転送路を意味するものとする。

最初に、キャッシュメモリの機能について解説する。ここで、キャッシュデータの置き換え方式について説明を加える。キャッシュメモリは高速で小容量のバッファメモリである。CPUなどの高速な処理速度をもつ上位装置と大容量メモリ装置などの低速なアクセス速度をもつ下位装置の間に設けられる。上位装置の高速な処理と下位装置の低速な処理との間のギャップを埋めるために用いられる。以下の説明では、上位装置をCPU、下位装置を主記憶装置とする。

CPUが主記憶装置をアクセスするとき、アクセスしたいデータの主記憶装置上の位置は局所性がある。これを空間的局所性という。また、同一位置のデータを繰り返しアクセスすることがある。これを時間的局所性という。CPUが主記

憶装置をアクセスするときに、当該データを含む適当な大きさのブロックを主記憶装置からキャッシュメモリへ転送しておく。データアクセスの局所性から、以後のCPUのメモリアクセスは高速なキャッシュメモリとの間で行うことができる。低速な主記憶装置をアクセスする必要がなくメモリアクセスを高速化できる。高速なアクセス速度をもつメモリデバイスを主記憶装置として用いればよいが、大容量のためコストが大きくなる。キャッシュメモリは高速でも小容量でありコストを低く抑えることができる。

キャッシュメモリ上に当該データが存在しない場合、これをキャッシュミスと呼ぶ。この場合には、当該データを含むブロックを主記憶装置からキャッシュメモリへ転送する。このとき、キャッシュメモリへ読み込まれるブロックが格納される領域を確保する必要がある。そのために、適当な規則によって選ばれたブロックがキャッシュメモリ上から追い出される。この追い出(置換)されるブロックが更新されている場合には、再び主記憶装置へ書き戻す必要がある。このように、キャッシュメモリと主記憶装置間でデータブロックが置き換えられる。キャッシュデータの置き換え方式として、ライトスルー方式とライトバック方式がある。

1) ライトスルー方式とは、CPUからキャッシュメモリへデータの書き込みが生じたときに、常にそのデータを主記憶装置へも書き込む方式である。このキャッシュメモリと主記憶装置の両方へ書き込む動作をライトスルーという。したがって、キャッシュミスで主記憶装置からブロックを読み込むときに、置換されるブロックが更新されていても、その更新データはすでに主記憶装置へ書き込まれており捨ててしまってよい。
2) ライトバック方式とは、上位装置からキャッシュメモリへデータの書き込みが生じても、そのデータを主記憶装置へ書き込まない方式である。したがって、キャッシュミスで主記憶装置からブロックを読み込むときに、置換されるブロックが更新されていれば、そのブロックを主記憶装置へ書き戻す必要がある。この書き戻しの動作をライトバックという。

以上がキャッシュメモリの説明である。一方、ディスクキャッシュとは、低速な処理速度のディスク装置と高速な処理速度の入出力チャネルとの間に設けられるキャッシュメモリである。上位装置は入出力チャネル(以下I/Oチャネルとも呼ぶ)、下位装置はディスク装置である。以下、従来のキャッシュメモリ付きディスク·サブシステムについて、その構成とキャッシュ動作を述べる。

3.1.1 従来のディスク·サブシステムの構成

図3.1中の左図に示されるのが従来のキャッシュメモリ付きディスク·サブシステムの構成[76]である。ディスク·サブシステムはディスク·コントローラとディスク·ユニットで構成される。ディスク·ユニットは複数のディスク装置を保持する。ディスク·コントローラはディレクタとキャッシュメモリで構成され

る。ディレクタはI/Oチャネルから来る入出力要求にしたがって、ディスク･ユニットとI/Oチャネル間のデータ転送を制御する。そのために、データ転送路であるパス毎にディレクタが設けられる。図の例では2個のディレクタをもつ。複数のディスク装置は、数本のパスでディスク･コントローラ中のディスクキャッシュと接続され、このパス(D-Cパス)を介してデータ転送を行う。ディスク装置は、接続された複数パスのいづれを使用してもデータ転送が行える。これをダイナミック･クロスパス[76][77]と呼ぶ。I/Oチャネルも数本のパスでディスク･コントローラ中のディスクキャッシュと接続され、このパス(以下C-Cパスと呼ぶことにする)を介してデータ転送を行う。図の例では、D-Cパス、C-Cパスはそれぞれ2本である。文献[76]に挙げられた実際に使用されているディスク･サブシステムでは、D-Cパス、C-Cパスはそれぞれ4本、8本である。図3.1中の右図に示されるのは、デュアル･フレーム構成と呼ばれるもので、左図に示される構成を2つ組みにして用いている。ディスク･コントローラ1がもつ2本のD-Cパスの一方はディスク･ユニット1と接続され、他の一方はディスク･ユニット2と接続される。同様に、ディスク･コントローラ2がもつ2本のD-Cパスの一方はディスク･ユニット1と接続され、他の一方はディスク･ユニット2と接続される。このような接続にすることで、2つのディスク･コントローラは、2つのディスク･ユニット内のどのディスク装置のデータもアクセス可能となる。したがって、2つのディスク･コントローラのどちらかが故障した場合にも、故障していないディスク･コントローラによってディスク･ユニット内のディスク装置をアクセスでき信頼性が向上する。

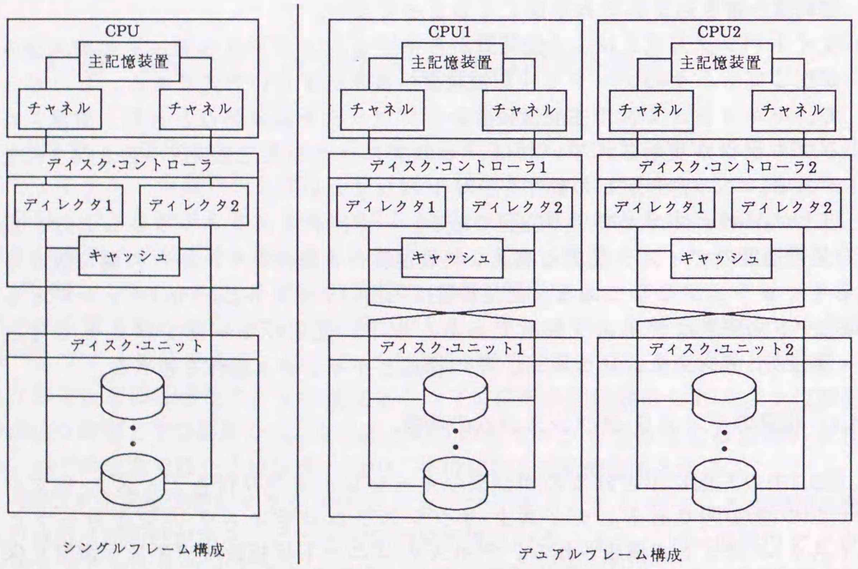


図 3.1 キャッシュメモリ付きディスク･サブシステムの構成

3.1.2 従来のディスク･サブシステムのキャッシュ動作

図3.2左図に示されるのは、ディスク装置とI/Oチャネル間のデータ転送をディスクキャッシュを用いて行った場合の基本動作である。図3.2右図はそのキャッシュ動作フローを示している。キャッシュデータの置き換え方式として、ライトスルー方式を採用した場合を表している。ディスク･コントローラが故障した場合にはそれが保持しているディスクキャッシュ内の情報も失われる。このとき、ライトバック方式では、上位装置から書き込まれた情報がディスク装置へ書き込まれないまま失われてしまう。したがって、ディスクキャッシュのキャッシュ動作としてはライトスルー方式が一般的である。ライトバック方式を採用する場合には、ディスクキャッシュをバッテリーバックアップ機構[55]などで不揮発メモリとする必要がある。以下動作を説明する。

ディスク装置とI/Oチャネルとの間のデータ転送はブロック単位で行われる。データ転送は4つの場合(リードヒット、リードミス、ライトヒット、ライトミス)に分けられる。リードヒットとは、I/Oチャネルからのアクセス要求がデータの読み込み要求で、その当該データがキャッシュに存在する場合である。リードミスとは、存在しない場合である。ライトヒットとは、I/Oチャネルから

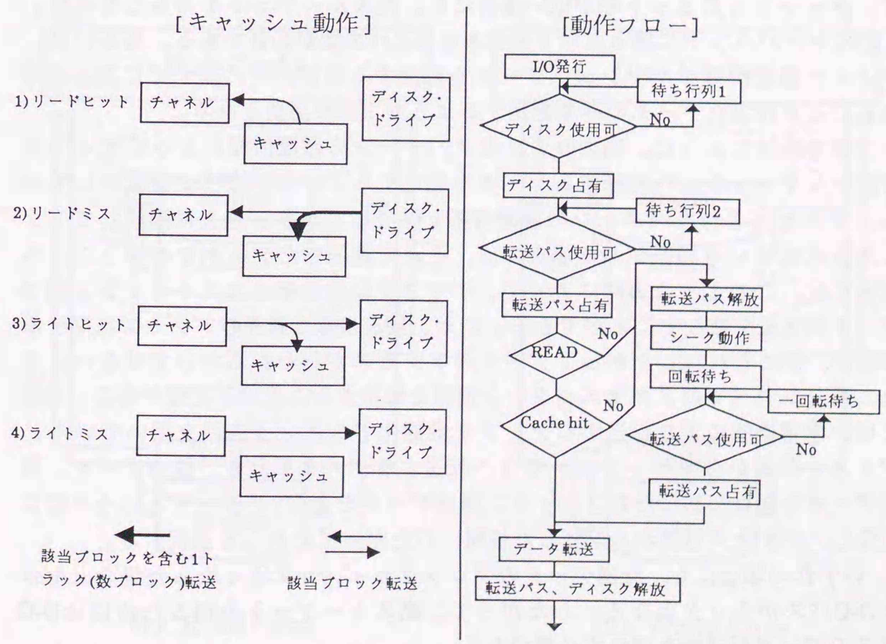


図 3.2 ディスクキャッシュのライトスルー方式におけるデータ転送と動作フロー

のアクセス要求がデータの書き込み要求で、その当該データがキャッシュに存在する場合である。ライトミスとは、存在しない場合である。

1) リードヒット:キャッシュに当該データが存在するので即座にI/Oチャネルへキャッシュからデータが転送される。
2) リードミス:キャッシュに当該データが存在しないので該当するディスク装置からデータをキャッシュへ転送し、同時にI/Oチャネルへも転送する。
3) ライトヒット:I/Oチャネルから該当するディスク装置へ更新データが転送される。キャッシュにも当該データが存在するのでこのデータを更新する。
4) ライトミス:キャッシュには当該データが存在しないので、I/Oチャネルから該当するディスク装置へのみ更新データが転送される。

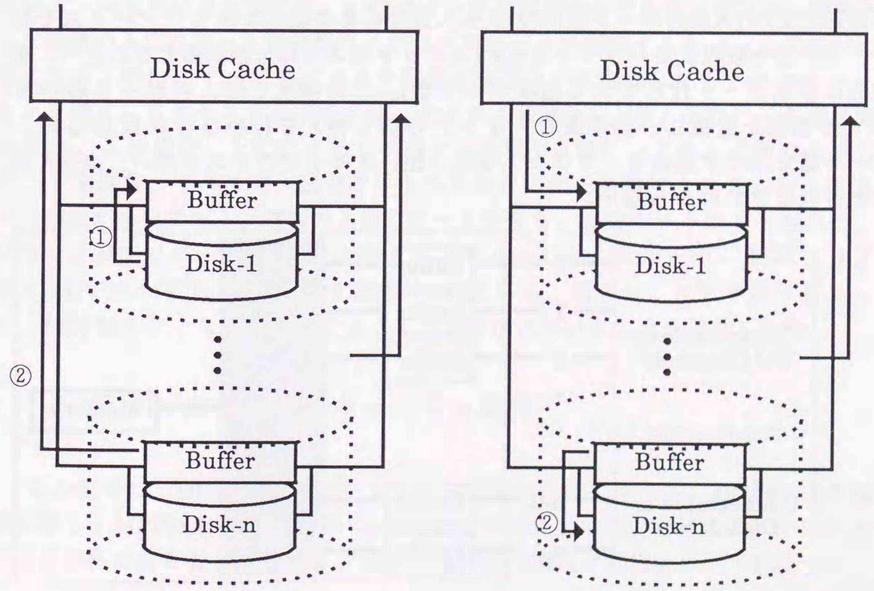
以上のように、キャッシュでリードミスが起きた場合には、ディスク装置から当該ブロックを転送する。データアクセスの局所性により、一般に、読み込まれたブロックの前後のブロックは、この後すぐに読み込まれる確率が高い。したがって、図3.2で示されるように、キャッシュでリードミスが起きた場合には、通常ディスク装置からキャッシュメモリへ当該ブロックを含む1トラック分(あるいは数ブロック分)のデータを転送する。ところが、キャッシュのヒット率が低い場合には、転送された1トラック分のデータ中当該ブロック以外は、この後読み込まれる割合が低く無駄なデータである。この無駄なデータ転送が頻繁に行われるため、これがオーバヘッドとなりスループットを悪化させる。したがって、キャッシュのヒット率が低い場合にも、高スループットを得るためには、上記のオーバヘッドに勝るだけ十分大きなD-Cパス数が必要である。あるいは、ディスク装置からキャッシュへデータを転送する時に、この後すぐに読み出されることが確実なデータのみを転送するような工夫が必要となる。

1章で述べたように、過去のアクセスパターンの履歴情報によってディスク装置からキャッシュへ転送するデータを指定するという方法[65]が提案されている。アクセスされたパターンの履歴情報を保持しておき、過去に連続してアクセスされたという履歴がある場合には、それら連続したデータをキャッシュへ転送する。このような方法によって、ディスク装置とディスクキャッシュ間のデータ転送量を減らすことができる。だが、履歴情報は過去のアクセスに関する情報で、常にそれにしたがってデータのアクセスが行われるわけではない。また、過去のすべてのアクセスパターン情報を保持することは不可能である。実際には、学習機能により転送するデータを最適化するという方法を用いている。ディスク装置からキャッシュメモリへ転送するデータ単位を、該当データ、該当データを含む連続した数ブロック、該当データを含む1トラックという単位に区分し、アクセス履歴から学習した情報にしたがってそれらを選択する。

いずれの場合にも、転送データのブロックサイズが大きくなった場合はやはりD-Cパスがネックとなる。したがって、高スループットを得るためにはD-Cパスのデータ転送幅を増やす必要がある。

3.1.3 RPSミスによる回転待ち

図3.2の右図の動作フローに示されるように、ディスク装置がシーク動作と回転待ちをし、データ転送が可能になったにも関わらず、D-Cパスが他のディスク装置のデータ転送中で使用可能なパスがない場合には、このディスク装置はデータ転送が行えない。これをRPSミスという。RPSミスが起こると、ディスク装置は、さらに一回転するのを待たなければ、データ転送を行えない。キャッシュのヒット率が低い場合には、ディスク装置へのアクセス回数が増え、D-Cパスの負荷が重くなる。これに伴いRPSミスが頻繁に生じ処理性能を低下させる。1章で述べたように、この点に関してはディスク装置内に小容量のバッファを設けたB-DISK[63][64]により解決されている。図3.3は、バッファ内蔵型ディスク装置を用いた場合のディスク装置とディスクキャッシュ間のデータ転送を示している。図3.3の(1)はディスク装置からデータを読み出す場合、図3.3の(2)はディスク装置へデータを書き込む場合を表している。データを読み出す場合にRPSミスが起きた時には、このバッファへデータを一旦転送しておく(図中①)。D-Cパスが空いた時点で、このバッファからデータの読み出しが行われる(図中②)。ディスク装置へデータを書き込む時も、D-Cパスが空いた時点で、このバッファへ一旦データを転送しておく(図中①)。ディスク装置が書き込み可能になった時



(1) Read Case　　(2) Write Case

図3.3 バッファ内蔵型ディスク装置のデータ転送

点で、このバッファからディスク装置にデータの書き込みが行われる(図中②)。これにより、RPSミスが起きても、一回転するのを待つ必要がなく、待ち時間を小さくでき処理速度の向上が望める。

上記のように、RPSミスは、多数(64台程度)のディスク･ドライブ装置が数本(4本程度)のD-Cパスを共用するために起こる。したがって、ディスク･ドライブ装置の台数と同数の専用のD-Cパスを持つことで、RPSミスを回避でき、性能の低下を抑えることができる。

3.1.4 従来のディスクキャッシュの構成

以上述べたように、従来型ディスク･サブシステムでは、ディスクキャッシュとディスク装置間のデータ転送がボトルネックとなっている。特に、ディスクキャッシュのヒット率が低い場合には、このボトルネックが顕著である。したがって、この間のデータ転送幅を増やさなければ、著しいスループットの向上は望めない。また、キャッシュのヒット率が高い場合にも高スループットを得るためには、C-Cパス数を増やす必要がある。D-CパスおよびC-Cパス数を増やす場合、従来型のディスク･サブシステムでは、ディスクキャッシュのデータ転送幅がネックになる。

図3.4に示されるのが、従来のディスクキャッシュの構成である。複数の転送パス(C-CパスとD-Cパス)に接続されるため多重ポート化される。(入出力ポートの転送速度)×(入出力ポート数)のアクセス速度をもつ高速なメモリデバイスを用い、速度差を吸収するバッファとn×1スイッチにより時分割制御することで、仮想的に多重ポート化している(nはポート数)。この構成では、各ポートのバッファの状態を監視し、効率良くスイッチの切り替えを行わなければならい。ポート数を増やす場合も、アクセス速度の速いメモリデバイスを用い、高速に制御を行わなければならない。

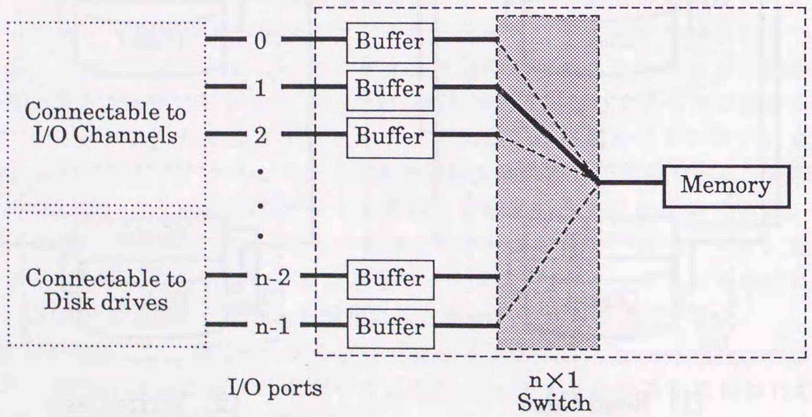


図3.4 従来のキャッシュメモリの構成例

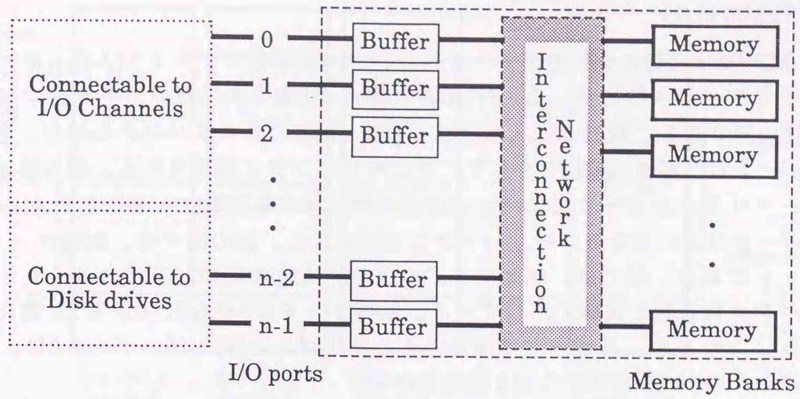


図3.5 複数バンクからなる多重ポートメモリの構成

ディスク装置からのデータ転送速度[76]は、4KBytes(4096Bytes)で1.3msecである。バイト転送する場合、320nsec/Byteの転送速度が必要である。8ポートのメモリを構成する場合には、40nsec/Byteの転送速度が必要となる。現在のDRAM(Dynamic Random Access Memory)のアクセス速度[78]は数十nsecであり、8ポート程度で限界となる。それ以上のポート数のメモリを構成する場合、メモリモジュールを複数用いる必要がある。図3.5に示されるように、複数のメモリモジュールを用いる場合には、n×1スイッチでは不十分であり、相互接続網を介して複数の入出力ポートと複数のメモリモジュールを接続するといった工夫が必要となる。特に、すべてのディスク装置が専用のパスを介してディスクキャッシュを独立にアクセスする構成では、ディスク装置数以上の入出力ポートが必要となる。この場合、上記のメモリ構成では実現不可能といえる。多数の入出力ポートを持ち、すべての入出力ポートから独立にデータのアクセスが行えるメモリ構成が必要である。複数の入出力ポートをもち、どのポートからも独立に、ブロック単位でデータの入出力が行えるメモリとして、マルチポート･ページメモリ(MPPM)がY.Tanakaにより提案[1]されている。著者は、以下で述べるように、MPPMをディスクキャッシュとして階層的に用いることを提案した(1)。

3.2節 DIMPの構成とキャッシュ動作

この節では、DIMPの構成とキャッシュ動作について述べる。DIMPの基本構成を示し、MPPMを階層的に用いることにより構成上の自由度が高く、システム拡張が容易であることを述べる。まず、MPPMの構成と動作について解説する。

3.2.1 マルチポート･ページメモリの構成と動作

[MPPMの構成]

MPPMは複数の入出力ポートを持ち、ページ単位でデータの入出力が行えるメモリデバイスである。図3.6に示されるのが8本の入出力ポートを持つMPPMの構成例である。MPPMは入出力ポート数と同数のメモリバンクを持つ。各入出力ポートは、相互接続網を介して、各メモリバンクと接続される。相互接続網はロータリネットワークである。相互接続網に与える制御コードにしたがって、各ポートは常に異なるメモリバンクと接続される。図の例では、制御コードは3ビットである。図では、制御コードの値が3(011_2)の時の、入出力ポートとメモリバンクの接続を実線で示している。制御コードは0から7、0から7と常に変化する。このとき、各ポートは順番に各メモリバンク(例えば、ポート0は、メモリバンクの0から7、0から7)と接続される。

図3.7に示したのは、ページデータの格納形式である。ページデータは、すべてのメモリバンクに対してインターリーブされて格納される。したがって、ページのサイズはメモリバンク数の倍数でなければならない。図の例では、メモリバンク数がmで、ページのサイズはm×nである。ページ内のi番目のデータは、((i-1) mod m)番のメモリバンクに格納される。ページの先頭のデータは常にメモリバンク0に格納される。したがって、j番目のポートは、制御コードが((m-j)mod m)の時にページデータのアクセスを開始する。

[MPPMの動作]

メモリバンク数mが8の場合でポート0を例にデータの入出力を説明する。ポート0がメモリバンク0と接続された時点で、データの転送が開始される。この時の制御コードは0である。制御コードが0から7へ一巡する間(これを1スライ

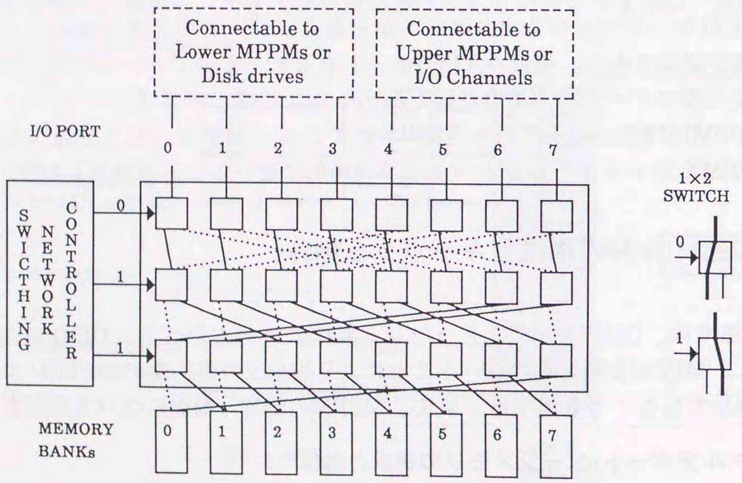


図3.6 MPPMの構成例

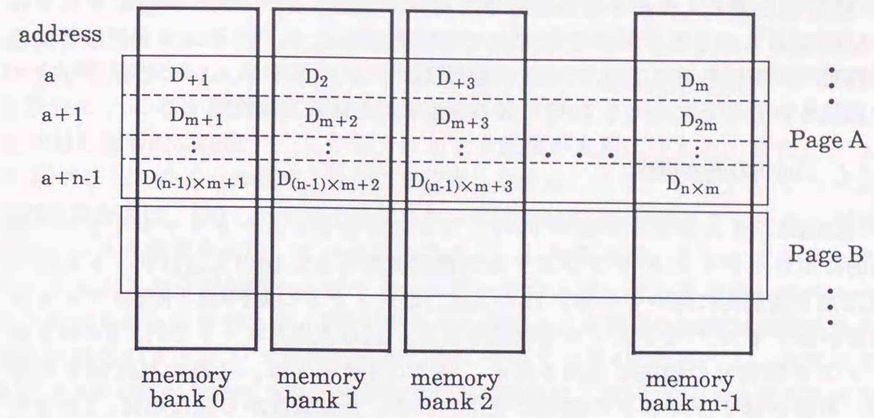


図3.7 MPPMのページデータの格納形式

スと呼ぶ)に、ポート0は、順にメモリバンクの0から7へ接続される。1スライスで、8個のデータをアクセスする。この間、すべてのメモリバンクへ、各々アドレスaが与えられる。次の1スライスでは、すべてのメモリバンクへ、アドレスa+1を与え、同様に8個のデータをアクセスする。これをnスライス目まで繰り返し、1ページの全データのアクセスが完了する。nスライス目でメモリバンクへ与えられるアドレスはa+n-1となる。このような処理が各ポートで独立に行われる。例えば、ポート1は制御コートが7の時に、メモリバンク0と接続されデータ転送が開始される。ポート1の場合の1スライスとは、制御コードが701~6と一巡する期間をいう。各ポートは、常に異なるメモリバンクをアクセスするので、アクセス衝突はない。同一のブロックであっても、異なるポートから同時に(厳密には、数クロックずれて)アクセス可能である。この様に、各入出力ポートはアクセス衝突なくページ単位でデータの入出力が行える。

各入出力ポートにおいて、データ転送はそのポートがメモリバンク0と接続された時点で開始される。したがって、MPPMのデータの入出力では、ポート切り替えによる待ち時間を必要とする。通常、これは入出力ポート数に比例する。しかし、転送ブロックサイズに比べて入出力ポート数が小さな場合には、著しいオーバヘッドではない。特に、ディスクキャッシュとして用いる場合には、転送ブロックサイズが大きいので問題とならない。ブロックサイズをB(KByte)、入出力ポート数をmとし、各ポートからバイト転送されるとき、ポート切り替えによる平均待ち時間はブロック転送時間×(m/(B×1024×2))となる。例えば、16ポートのMPPMで、4Kバイト・ブロックを転送する場合では、ブロック転送時間の512分の1がポート切り替えによる平均待ち時間となる。128ポートのMPPMで、4Kバイト・ブロックを転送する場合では、ブロック転送時間の64分の1がポート切り替えによる平均待ち時間となる。このように、MPPMでは小さなオーバヘッドで入出力ポート数を増やすことができる。

MPPMをディスクキャッシュとして用い、複数あるディスク装置のそれぞれをMPPMの入出力ポートのそれぞれに接続することで、ディスク装置数と同数のD-Cパスが得られる。各ディスク装置は、ディスクキャッシュと専用のパスで接続されることになり、RPSミスの発生を抑えることもできる。

3.2.2 DIMPの基本構成

図3.8に示すのがマルチポート･ページメモリをディスクキャッシュとして階層的に用いたディスク･サブシステムの構成例である。16の入出力ポートをもつMPPMを階層的に用いている。16の入出力ポートをもつMPPMと8台のディスク装置からディスク･モジュールを構成できる。16の入出力ポートの内、8ポートがディスク装置との接続に使用される。残りの8ポートは、上層のMPPMとの接続、あるいはチャネルとの接続に使用される。上位装置から見た場合、8台分のディスク容量があり、8本の入出力パスをもつディスク装置と同様に扱うことができる。さらに、16の入出力ポートをもつMPPMと先のディスク･モジュール8個から新たにディスク･モジュール(64台分のディスク容量があり、8本の入出力パスをもつ)を構成できる。16の入出力ポートの内、8ポートがディスク･モジュールとの接続にそれぞれ使用される。残りの8ポートは、上層のMPPMとの接続、あるいはチャネルとの接続に使用される。MPPMは、複数の入出力ポートをもち、アクセス衝突なくブロック単位でデータの入出力が行えるメモリデバイスである。MPPMをディスクキャッシュとして用いることにより、各ディスク･ドライブは、衝突なくキャッシュメモリへアクセスできる。同様に、各ディスク･モジュールは、上位のキャッシュメモリへ衝突なくアクセスできる。

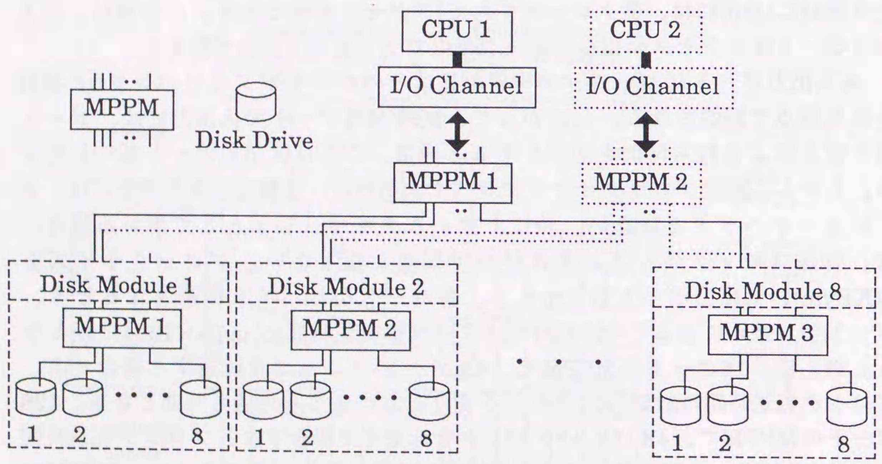


図3.8 マルチポート･ページメモリをディスクキャッシュとして階層的に用いたディスク･サブシステムの構成

また、図3.8の破線で示したように、上層のMPPMをさらに1個用いることで、チャネルと接続されるパス数を増やすことができる(先述したデュアル･フレーム構成である)。このように、比較的少ない数の入出力ポートを持つMPPMを用いて、これを階層的に構成したディスク･サブシステムがDIMPである。MPPMを階層的に構成することで、システム拡張が容易に行える。また、デュアル･フレームのような構成上の工夫を適用しやすい。ただ、デュアル･フレーム構成の場合には、I/Oチャネルと接続される独立なMPPMが2個あり、それらが複数のディスク装置を共用していることになる。したがって、キャッシュデータのコヒレンシの問題が残る。キャッシュデータのコヒレンシを考慮したキャッシュ動作を行う必要がある。I/Oチャネルと接続される独立なMPPMが複数ある構成の場合のキャッシュ動作と処理性能については第4章で述べる。本章の第3.3、3.4節で述べるキャッシュ動作および処理性能は、I/Oチャネルと接続されるMPPMが1つの構成の場合を対象する。

また、図3.8に示したDIMPの構成は、二階層の構成であるが、種々の構成が考えられる。図3.9に示したように、128の入出力ポートをもつMPPMを1個用いた一階層の構成も考えられる。64の入出力ポートを64台のディスク装置のそれぞれと接続し、残りの64の入出力ポートをチャネルとの接続パスとして用いる。ただ、システム拡張が容易に行え、構成上の工夫を適用しやすいという点では、図3.8に示した構成が良いと思われる。

[キャッシュ容量と転送ブロック単位]

いずれの階層構成の場合にも、ディスク装置と接続される一番下層のMPPMのメモリ容量を大きくし、これにディスクキャッシュの機能をもたせる。従来のディスクキャッシュと同程度のキャッシュ容量を仮定すると、数GBの容量のディスク装置を用いる場合、図3.8の構成の最下層のMPPM(8台のディスク装置が接続される)のメモリ容量は数十MBとなる。また、上層のMPPMもキャッシュとして機能するが、これを挟む上･下層のMPPMとのデータ転送のためのバッファとしての機能を期待しており、比較的小さな容量で良い。データ転送単位についても、キャッシュミスが起きてディスク装置から最下層のMPPMへデータを読み込む場合のみトラック単位のデータ転送とし、他のデータ転送はすべてブロック単位とする。

3.2.3 ポート切り替えにより起こる待ち時間の解消

DIMPはMPPMによる階層構造を成す。先述したように、MPPMのデータの入出力はバンク番号が0のメモリバンクを基準に行われる。よって、MPPM同士の入出力ポートの接続では、間にバッファを入れる必要がある。しかし、MPPM間の入出力ポートの接続関係を考慮して、同期した制御コードをそれぞれのMPPMのスイッチング･ネットワークに与えることで、バッファを入れなくて

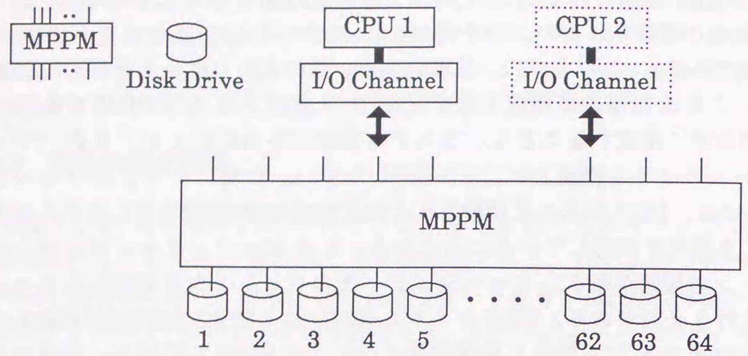


図3.9　マルチポート-ページメモリをディスクキャッシュとして用いたディスク-サブシステムの1階層構成

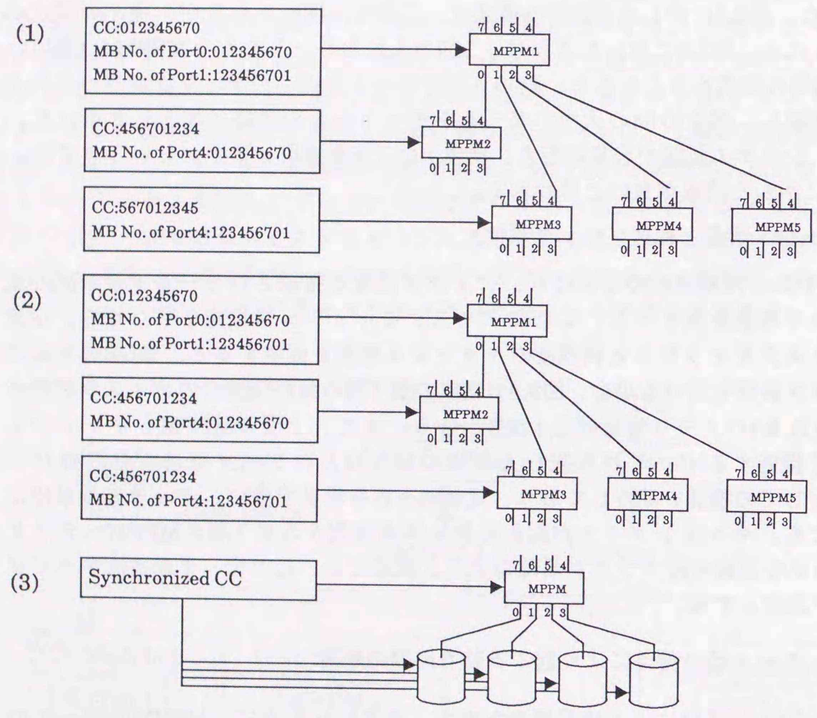


図3.10　MPPMのポート接続と制御コード

もよい。また、バッファでの待ち時間を解消できる。図3.10に示した、3つの場合について説明する。

(1)これは上層のMPPM1の各ポートと接続される下層のMPPM(MPPM2,MPPM3…)のポート番号を統一した場合である。上層のMPPM1

のポート0,1,2,…に下層のMPPM2,MPPM3,MPPM4,…のポート4がそれぞれ接続される。上層のMPPM1の制御コードを012…と与えた場合、ポート0が接続するMPPM1内のメモリバンク番号は012…となる。ポート0が接続されている下層のMPPM2のポート4が接続するMPPM2内のメモリバンク番号も012…でなければならない。したがって、MPPM2へ与える制御コードは456…となる。同様に、上層のMPPM1の制御コードを012…と与えた場合、ポート1が接続するMPPM1内のメモリバンク番号は123…となる。ポート1が接続されている下層のMPPM3のポート4が接続するMPPM3内のメモリバンク番号も123…でなければならない。したがって、MPPM3へ与える制御コードは567…となる。以下同様に、MPPM4へ与える制御コードは670…となり、MPPM5へ与える制御コードは701…となる。

(2) これは下層のMPPM(MPPM2,MPPM3…)に与える制御コードを統一した場合である。(1)と同様に上層のMPPM1のポート0は下層のMPPM2のポート4に接続される。上層のMPPM1のポート1は下層の別のMPPM3のポート5に接続される。以下同様に、上層のMPPM1のポート2,3…は下層のMPPM4のポート6、下層のMPPM5のポート7に接続される。

(3) ディスク装置からその上層のMPPMへのデータ転送は、1トラック毎に行われるものとする。よって、各ディスク装置を同期して回転させ、これとMPPMへ与える制御コードを同期させることにより、小さなバッファ容量(小さな待ち時間)でデータ転送を行うことが可能であると思われる。ただ、ディスク台数が多い場合には制御が難しくなると思われる。

以上の動作は、下層のMPPMの入出力ポート数と上層のMPPMのポート数が同一の場合を例に説明されている。その他の場合には、下層のMPPMのポート数が上層のMPPMのポート数の倍数である、あるいは、上層のMPPMのポート数が下層のMPPMのポート数の倍数であるという条件が必要となる。また、転送ブロックサイズは上層と下層のMPPMの入出力ポート数の大きいほうの値の倍数である必要がある。

3.3節　DIMPのキャッシュ動作

DIMPではMPPMをディスクキャッシュとして用いる。一般のキャッシュメモリと同様に、キャッシュデータの置き換え方式として、ライトスルー方式とライトバック方式が考えられる。ここでは、それぞれの方式のキャッシュ動作について説明する。それぞれの場合の評価性能については後の3.4節で示す。キャッシュデータの置き換え方式がライトバック方式の場合には、キャッシュメモリはバッテリバックアップなどが施された不揮発メモリであり、ディスク・コントローラが故障してもキャッシュ内のデータは失われないものとする。

3.3.1　ライトスルー方式

図3.11に示すのがライトスルー方式の基本動作である。

1) リードヒットの場合：当該データがMPPM内に存在するので、MPPMから即座にデータの読み出し(図中①)が行える。
2) リードミスの場合：当該データがMPPM内に存在しないので、下層のディスクあるいは下層のMPPMから当該ブロックを読み込み(図中①)、一旦MPPMにデータを書き込む。これと同時にデータの読み出し(図中②)が行える。ディスクから読み込む場合には、当該データを含む1トラック分を読み込む。
3) ライトヒット、ライトミスの場合：必ずディスク装置へ当該データを書き込む必要がある。まず、一旦MPPMにデータを書き込む(図中①)。その後、下層のディスクあるいはMPPMにデータを書き込む(ヒット・ミスに関係なく、書き込みデータは、必ず一旦MPPMに書き込まれる)(図中②)。下層のMPPMに対するポートが使用中の場合には、ポートが空くのを待った後この処理を行う。この操作が繰り返えされて、書き込みデータは次第に下層に移動し、

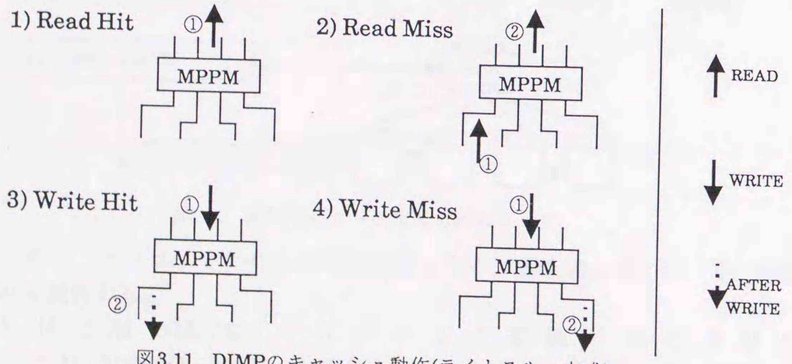


図3.11　DIMPのキャッシュ動作(ライトスルー方式)

最後にディスク装置へ書き込まれる。ディスク装置に対する書き込みは、当該ブロックのみである。書き込みデータが移動した通り道となるMPPMには、LRU(Least Recently Used)規則により捨てられるまで、このデータが残り続ける。したがって、書き込みデータが下層へ移動中に、このデータに対する読み出し要求が来た場合でも、上層のMPPMから即座に読み出しが行える。

LRU規則とは、キャッシュメモリから追い出されるブロックを決定する規則の1つである。キャッシュメモリへ格納されたブロック中、最も昔にアクセスされたブロックは、この後アクセスされる確率が最も低いであろうという統計的な解釈がある。LRU方式とは、最も昔にアクセスされたブロックをキャッシュメモリから追い出すブロックとして決定する方式である。LRU方式は最適な方式であると考えられ一般によく使われる。

3.3.2　ライトバック方式

図3.12に示すのが、ライトバック方式の基本動作である。

1) リードヒットの場合：当該データがMPPM内に存在するので、MPPMから即座にデータの読み出しが行える(図中①)。
2) リードミスの場合：当該データがMPPM内に存在しないので、下層のディスク装置あるいは下層のMPPMから当該データを読み込まなければならない。そのためには、要らないブロックをMPPMから追い出す必要がある。まず、LRU規則にしたがいブロックを追い出す。このとき、これが更新されたブロックならば、下層のディスク装置あるいは下層のMPPMに書き戻す(図中①)。その後、下層のディスクあるいは下層のMPPMから当該データを読み込む(図中②)。ディスク装置から上層のMPPMへのデータ転送は、当該ブロックを含む1トラック毎に行われる。これと同時に、MPPMから当該データの読み出しが行える(図中③)。これらの処理は各層のMPPMで独立に行われる。

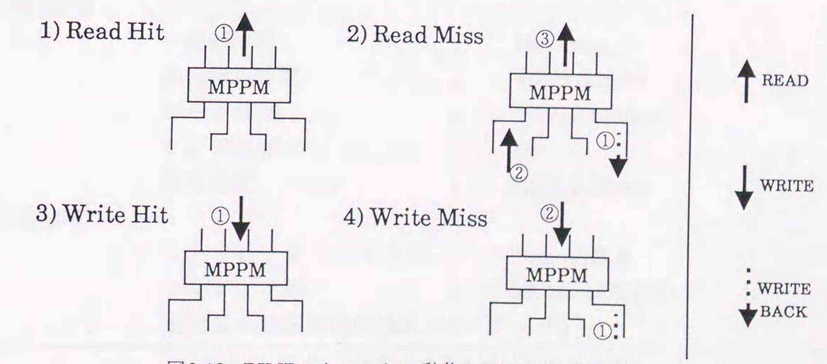


図3.12　DIMPのキャッシュ動作(ライトバック方式)

3) ライトヒットの場合：当該データがMPPM内に存在するので、MPPMに対して即座にデータの書き込みが行える(図中①)。
4) ライトミスの場合：当該データがMPPM内に存在しないので、更新データを格納するために要らないブロックをMPPMから追い出す必要がある。まず、LRU規則にしたがいブロックを追い出す。このとき、これが更新されたブロックならば、下層のディスク装置あるいは下層のMPPMに書き戻す(図中①)。ディスク装置に対する書き込みは、更新された当該ブロック毎に行われる。その後、MPPMへデータの書き込みが行える(図中②)。

3.3.3 固定長レコードと可変長レコード

ワークステーション等の小型コンピュータでは、ディスク装置等のI/O機器との接続のためにSCSI(Small Computer System Interface)バスが用いられる。一般に、SCSIバスに接続されたディスク装置では、固定長レコードを扱う。固定長レコードの場合、ディスク装置に格納される領域が固定されており、データの書き込みに際して、そのデータのディスク上での領域が存在しないということはない。一方、汎用大型機と入出力チャネルを介して接続されるディスク･サブシステムでは、CKD(Count Key Data)フォーマット[76]と呼ばれる可変長レコードを扱うものがある。この場合には、ディスク装置に格納される領域が不定であり、ディスク上に存在しない新しいデータの書き込みに際しては、まず、ディスク上に領域を確保する必要がある。よって、常にキャッシュにデータを書き込むという動作が行えない。したがって、CKDフォーマットとよばれる可変長レコードの場合には、上述したキャッシュ動作をそのまま用いることができない。最近では、制御が簡単なため固定長レコードを扱うディスク装置が多い。上述したキャッシュ動作は固定長レコードを扱うディスク装置を対象として考えている。本論文では、可変長レコードを扱うディスク装置に関しては議論しない。

3.4節 DIMPの処理性能

本節では、シミュレーションにより得られた処理性能を示し、DIMPの性能と従来のディスク･サブシステムの性能を比較し考察を加える。まず、従来のディスク･サブシステムの処理性能を示し考察する。前記3.1節で述べたように、従来のディスク･サブシステムではD-Cパスがボトルネックとなっていることを示す。その後、DIMPの処理性能を示し考察を加える。シミュレーションは、待ち行列モデルを基本とした性能評価シミュレータCABを用いて行った。

3.4.1 従来型ディスク･サブシステムの性能評価

以下では、ディスク装置の性能とシミュレーション条件を示す。

(1) ディス装置の性能とデータ転送時間

表3.1に、シミュレーション対象とするディスク装置の性能とデータ転送時間を示す。これらの値は、参考文献[76]中に示されているもので、実際に使用されているディスク装置の性能である。したがって、得られるシミュレーション結果は現実的な値である。ディスク装置にアクセスする場合、常にシーク動作が行われるわけではない。同一トラック上のデータであれば、シーク動作は行われない。表中のランダム度とは、シーク動作が行われるか否かを確率として与えたものである。文献[46]で述べられているように、この値は通常1/3となる。本性能評価でも同様の値を用いた。4Kバイト/ブロック･データのディスク装置からの転送時間は1.3msecとなる。これに回路の遅延時間を含めて、1ブロックデータの転送時間を3msec/ブロックとした。また、1トラック･データのディスク装置からの転送時間は16.7msecとなる。これに回路の遅延時間を含めて、1トラック･データの転送時間を18.4msec/トラックとした。

[回転速度]			
	一回転時間	:	16.7msec
	回転待ち時間	:	平均8.3msec
	シーク時間	:	平均12.5msec
	ランダム度	:	1/3
	転送速度	:	3,000KB/sec
[転送時間]			
	リードヒット、ライト時	:	3msec/BLK
	リードミス時	:	18.4msec/TRK
	(1BLK＝4KB固定/1TRK＝1トラック)		

表 3.1 ディスク装置の性能とデータ転送時間

(2) シミュレーション条件

表3.2に、シミュレーション条件を示す。実際に使用されているディスク･サブシステムの入出力リクエストのトレース･データを用いることで、より現実的に性能評価が行える。しかし、そのようなデータを得ることができないため、入出力リクエストは確率分布に従うものとした。一般に、ディスクアクセス要求はライト要求よりもリード要求の割合が高い。したがって、リード:ライトの比を4:1固定とした。入出力リクエストの発行件数を20,000件としてシミュレーションを行った。

(1) I/Oの到着間隔はランダム(指数分布)である。
(2) リード:ライト比を4:1固定とする。
(3) 転送ブロック長を4KByte固定とする。
(4) リードミス時には、当該データを含む1トラックを読み込むものとする。
(5) キャッシュとチャネル間のデータ転送時間も3msec/ブロックとする。

表 3.2 シミュレーション条件

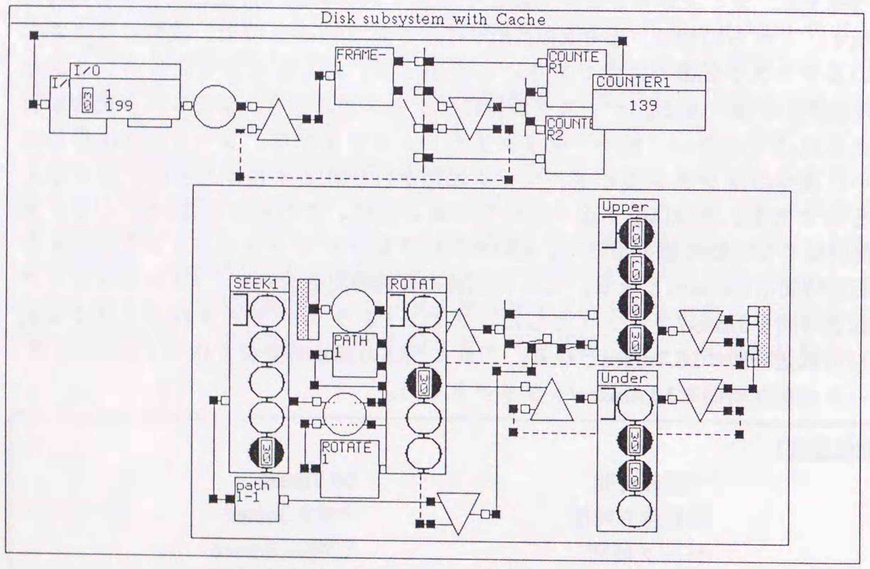


図3.13 CABのEdit-Window画面ハードコピー
(従来のキャッシュ付きディスク･サブシステムの動作を定義)

上述したように、シミュレーションは先の2章で述べた待ち行列網シミュレータCABを用いて行った。図3.13は、CABのEdit-Windowの画面ハードコピーである。Edit-Window内に表示されているモデルは、従来のキャッシュ付きディスク･サブシステムの動作を定義したものである。この動作は前記3.1節の図3.2の動作フローに示されたものである。図3.2に示される動作フロー図中の処

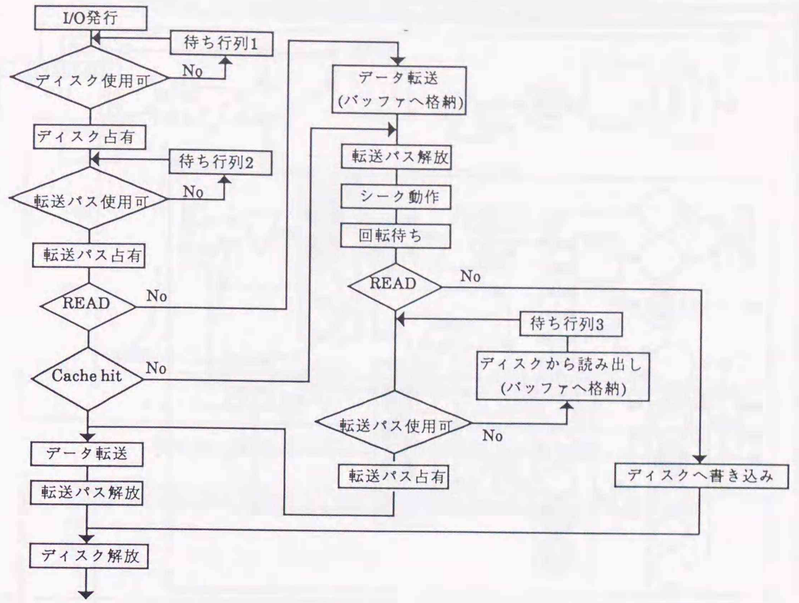


図3.14 バッファ内蔵型ディスク装置を用いたディスク･サブシステムの動作フロー

理時間要素をQUEUEING-MODELのSERVERに割当て、処理時間をそのサービス時間として設定する。また、種々のTOKENの動作フロー制御用モデル(CONNECTOR,RESERVER,RELEASER,TRANSISION,PLACE)を用い、図3.2に示される動作フロー図と同様にTOKENが移動するようそれらを構成する。このようにして、対象システムの動作の定義が行われる。TOKENの生成モデルであるTOKEN-PRODUCER(図中I/Oとラベル付けされているモデル)の確率分布パラメータを設定することにより、単位時間当たりに発行される入出力リクエスト数が決められる。C-Cパス数やD-Cパス数およびそこでの処理時間は、対応するQUEUEING-MODEL(Upper-PathおよびUnder-Pathとラベル付けされている)のSERVER数やそのサービス時間を設定することにより決められる。これらパラメータの設定はメニュー形式で簡単に行うことができる。また、図3.14および図3.15は、それぞれバッファ内蔵型ディスク装置を用いた場合のディスク･サブシステムの動作フローおよびその動作を定義したCABのEdit-Window画面ハードコピーである。バッファ内蔵型ディスク装置を用いた場合のディスク･サブシステムについても、図3.14に示される動作フローにしたがって、図3.15に示されるように各基本モデルを配置し構成することで、その動作のモデリングが可能である。以上のように、CABは異なる動作機能をもつ種々の基本モデルを用意しており、CABを用いることによって、全くプログラミングすることなく対話的にシミュレーション実験が行える。

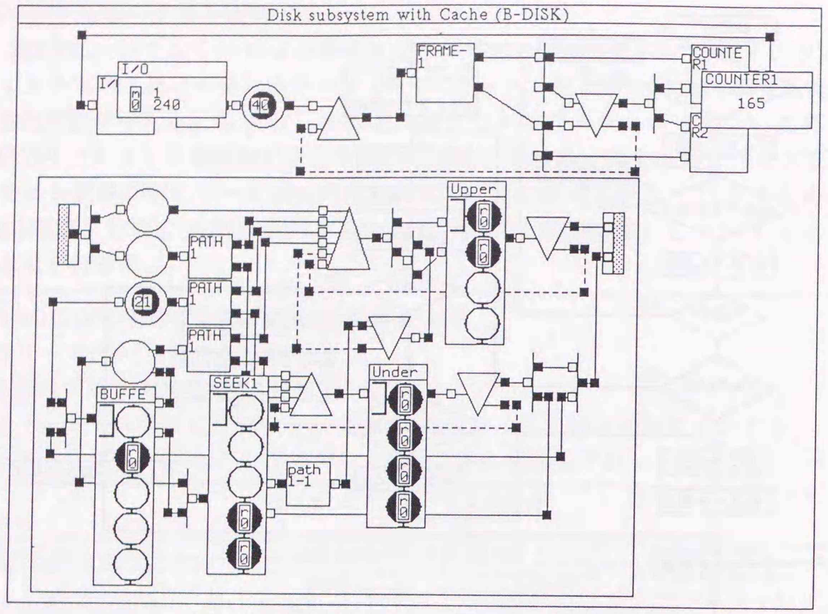


図3.15　CABのEdit-Window画面ハードコピー
(バッファ内蔵型ディスク装置を用いたディスク･サブシステムの動作を定義)

(3)　評価結果

上述したシミュレータによりシミュレーションを行い、単位時間当たりの発行I/O数と応答時間(TAT:Turn Around Time)の関係を求めた。ディスクキャッシュ(以下DCとも呼ぶ)のみをもち、ディスク･ドライブ数64台、D-Cパス数4、チャネルパス数8のディスク･サブシステムを構成1とする。構成1に対して、D-Cパス数を8にしたディスク･サブシステムを構成2とする。DCとバッファ内蔵型ディスク装置(以下B-DISKと呼ぶ)の両方を備え、ディスク･ドライブ数64台、D-Cパス数8、チャネルパス数8のディスク･サブシステムを構成3とする。図3.16、図3.17、図3.18は、それぞれ構成1、2、3について、キャッシュのヒット率を25、45、65、85%と変えた結果のグラフである。

(4)　考察

図3.16、図3.17、図3.18より、どの構成においても、キャッシュのヒット率が低い場合には、スループット(単位時間当たりのI/O数)が著しく悪いことが分かる。表3.3は、応答時間(TAT)が50msecの時のスループットを、図3.16、図3.17、図3.18のグラフから比較したものである。どのヒット率においても、D-Cパス数が4(表中[1])の場合に比べてD-Cパス数が8(表中[2])の場合は、ほぼ2倍のスループットが得られている。したがって、ディスク･ドライブとディスク･キャッシュ間のデータ転送がボトルネックであると分かる。また、D-Cパス数が8でB-

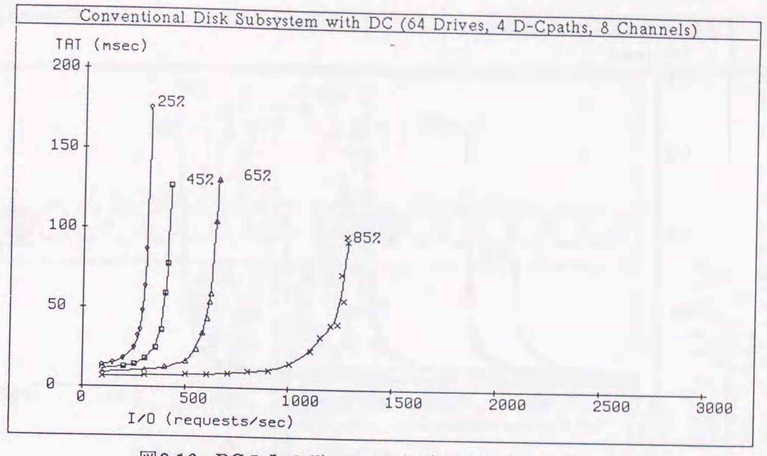


図3.16　DCのみのディスク･サブシステムの性能

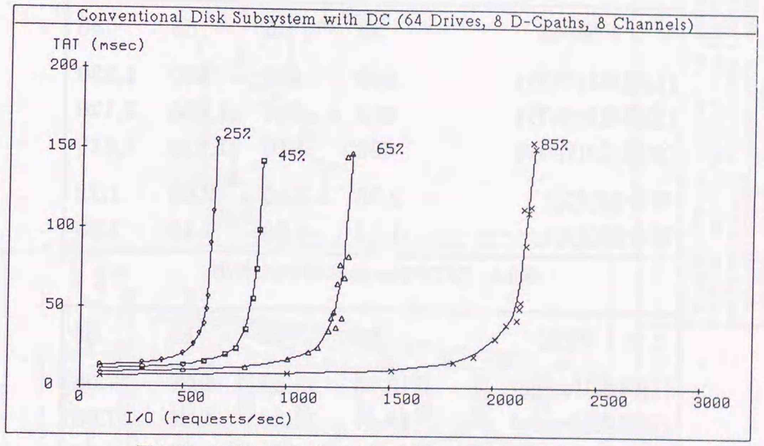


図3.17　DCのみのディスク･サブシステムの性能

DISKを用いない(表中[2])場合と、D-Cパス数が8でB-DISKを用いた(表中[3])場合を比べてみると、スループットは、10%程度しか向上していない。以上から分かるように、スループットを向上させるためには、ディスク･ドライブとディスク･キャッシュ間のデータ転送幅を増やす必要がある。

表3.4は、各構成について、低いトランザクション域(スループットが100件/秒)での応答時間を示したものである。応答時間は、キャッシュのヒット率が大きくなるにしたがって、値が小さくなっている。B-DISKを用いた場合には、ディスク装置に対するデータ転送をディスクの回転とは非同期に行うことができ、B-DISKを用いない場合に比べて応答時間が良くなっているのが分かる。

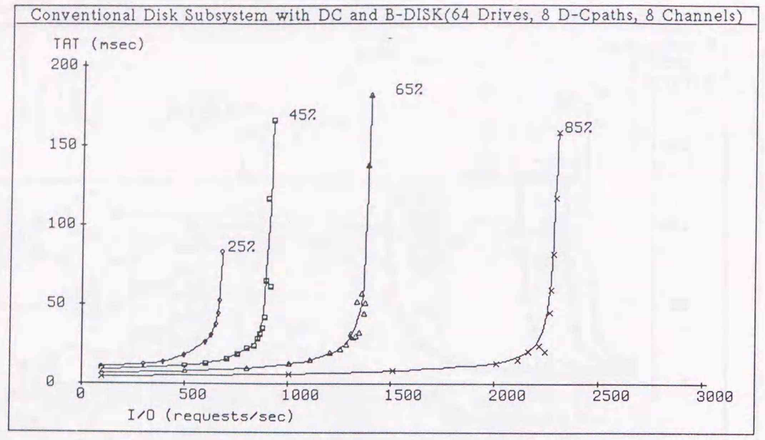


図3.18　DCとB-DISKのディスク･サブシステムの性能

ヒット率(%)	25	45	65	85
[1]構成1(件/秒)	300	400	620	1,230
[2]構成2(件/秒)	620	840	1,260	2,120
[3]構成3(件/秒)	690	910	1,410	2,310
割合([2]/[1])	2.06	2.10	2.03	1.72
割合([3]/[2])	1.11	1.08	1.11	1.09

表3.3　TATが50msecの時のI/O件/秒

ヒット率(%)	25	45	65	85
[1]構成1(msec)	13.86	11.53	9.30	7.08
[2]構成2(msec)	13.47	11.38	9.30	7.08
[3]構成3(msec)	11.07	8.98	6.76	4.66

表 3.4　I/Oが100件/秒の時のTAT

3.4.2　DIMPの性能評価

(1)　ディスク装置の性能とシミュレーション条件

ディスク装置の性能およびデータ転送速度は表3.1と同じである。MPPM間のデータ転送もすべて、データ長を4KByte/ブロック(転送時間3msec)固定とした。シミュレーション条件も表3.2と同じである。シミュレーションに用いたDIMPの構成は、先の図3.8に示したものである。ディスク台数が64台、I/Oチャネル数が8で、従来型のディスク･サブシステムのシミュレーションに用いたモデルに対応するものである。I/Oの発行件数を50,000件としてシミュレーション

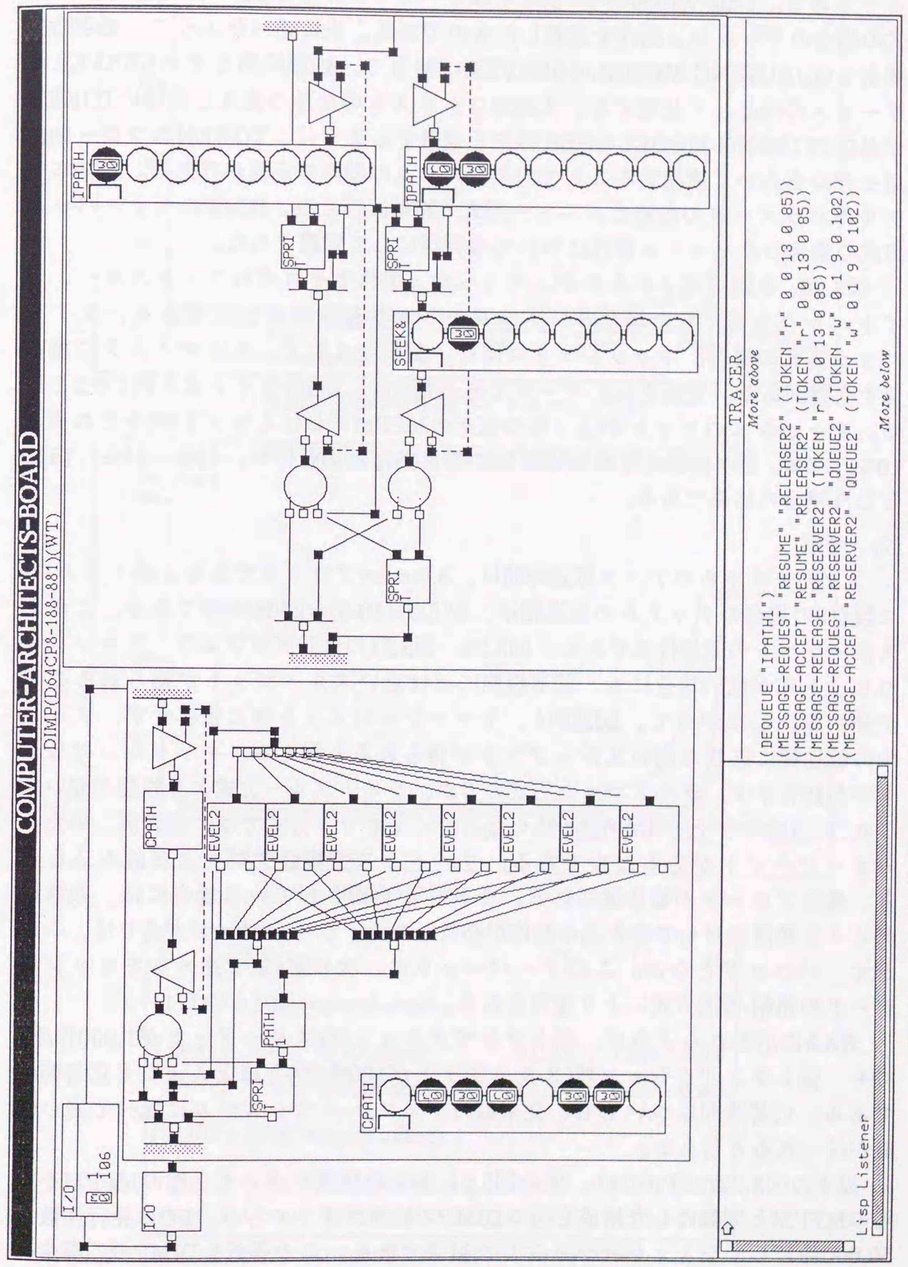


図3.19　CABのEdit-Window画面ハードコピー(DIMPのライトスルー方式の動作を定義)

を行った。図3.19は待ち行列網シミュレータCABのEdit-Windowの画面ハードコピーである。Edit-Window内に表示されているモデルはDIMPのライトスルー方式の場合のキャッシュ動作を定義したものである。先に述べたように、処理時間要素をQUEUEING-MODELのSERVERに割当て、処理時間をそのSERVERのサービス時間として設定する。入出力リクエストの処理の流れと同様にTOKENが各QUEUEING-MODELのSERVERを通過するように、TOKENのフロー制御用モデルを用いて構成することで対象システムの動作の定義が行える。各基本モデルのパラメータの設定はメニュー形式で簡単に行える。DIMPのライトバック方式の場合のキャッシュ動作についても同様にして定義された。

図3.20、3.21に示されるのが、キャッシュ動作をそれぞれライトスルー、ライトバック方式にした場合のシミュレーション結果のグラフである。ライトバック方式の場合、キャッシュミス時に、更新されたデータはディスクに書き戻す必要がある。更新されたデータである割合は、I/Oのライト比と同じ0.2とした。キャッシュのヒット率(上下層の個々のMPPMにおけるヒット率)をそれぞれ10%、30%、50%(1個の等価なMPPMに置き換えた場合には、19%、51%、75%)とした場合の結果である。

(2) 考察

入出力チャネルのデータ転送時間は、3.0msec/ブロックである。チャネル・パス数が8の場合のチャネルの転送幅は、8/(3.0×10-3)＝2666件/秒である。これがスループットの限界性能である。図3.20、図3.21の性能グラフより、キャッシュのヒット率が低い場合にも、限界性能にほぼ近いスループットが得られることが分かる。したがって、DIMPは、キャッシュのヒット率に依存せず、チャネルの転送幅に応じた高いスループットが得られると言える。キャッシュのヒット率が19%では、ライトバック方式の方が、ライトスルー方式より性能が悪い。これは、I/Oのライト比が0.2と低くく、ライトバック方式では、書き戻しのためのオーバヘッドがあるためである。リードミスで複数ブロックを読み込むとき、置換ブロックが複数選ばれる。それらが更新されている場合には、複数のディスク装置に対して書き込み動作が必要となる。ライトバック方式では、これがオーバヘッドとなる。このオーバヘッドは、次の第4章で述べるキャッシュデータの格納・置換方式により改善される。

表3.5に示されるように、高トランザクション域(スループットが1,000件/秒)でも、低トランザクション域(スループットが100件/秒)とほとんど同じ応答時間である。応答時間についても、従来型のディスク・サブシステムに比べて良い性能が得られると言える。

以下の図3.22に示すのは、先の図3.8に示した構成において上層のMPPMを下層のMPPMと同数にした構成を持つDIMPの性能グラフである。I/Oの発行件数を50,000件としてシミュレーションした結果である。この構成をDIMPの二層全接続型構成と呼ぶ。この構成の場合には、上層のMPPMに関してキャッシュデータ

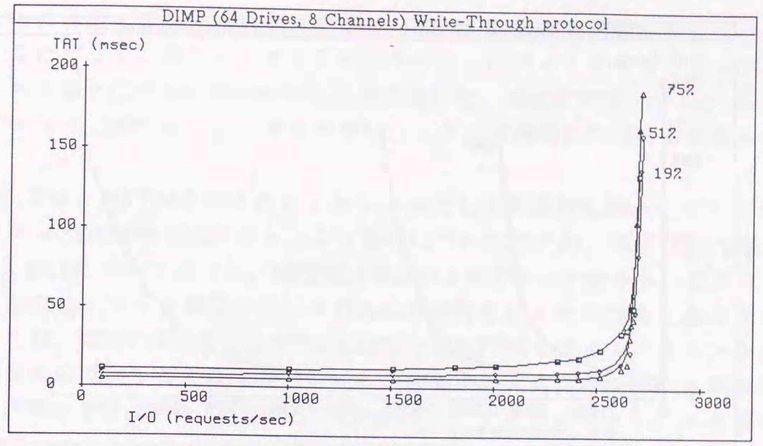


図3.20 DIMPの性能(ライトスルー方式)

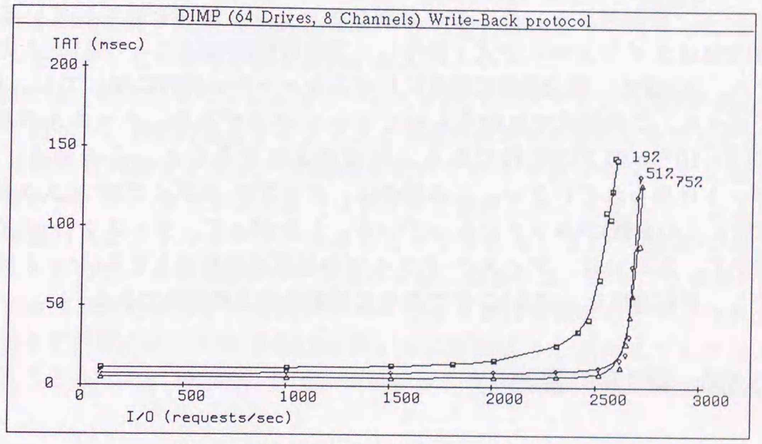


図3.21 DIMPの性能(ライトバック方式)

ヒット率(%)	19	51	75
(1) I/O＝100(requests/sec)			
・ライトスルー方式(msec)	11.11	7.90	5.49
・ライトバック方式(msec)	11.14	7.92	5.48
(2) I/O＝1000(requests/sec)			
・ライトスルー方式(msec)	11.59	8.18	5.71
・ライトバック方式(msec)	11.72	8.15	5.65

表3.5 I/Oが100件/秒と1000件/秒の時のTAT

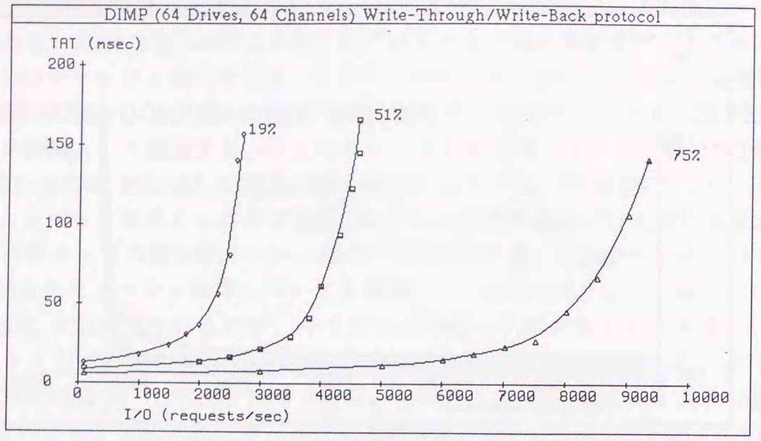


図3.22　DIMPの性能(二層全接続型構成)

のコヒレンシ問題がある。そこで、特別なキャッシュ動作を採用している。上層のMPPMはライトスルー方式で動作し、下層のMPPMはライトバック方式で動作する。DIMPの二層全接続型構成におけるキャッシュ動作に関しては、4章で詳しく述べる。この構成では64本ものチャネル･パスがあり、チャネルの転送幅は$64/(3.0\times10^{-3})=21{,}333$件/秒である。性能グラフを見ると、この値までのスループットは得られていない。この場合は、ディスク･ドライブ(ディスク装置の処理性能とその台数)がネックになっている。したがって、ディスク･ドライブ台数を増やす、あるいは、ディスク･ドライブの処理速度を向上する(シーク時間を小さくし、回転速度を上げる)ことでさらに性能の向上が期待できる。

3.5節　第3章の要約

本章では、マルチポート･ページメモリ(MPPM)をディスクキャッシュとして用いたディスク･サブシステム：DIMPのアーキテクチャについて述べた。

3.1節では、従来のキャッシュメモリ付きディスク･サブシステムの構成とキャッシュ動作について述べた。また、キャッシュメモリの機能についても解説した。従来のキャッシュメモリ付きディスク･サブシステムでは、ディスク装置とディスクキャッシュ間のデータ転送パスがボトルネックとなっていることを述べた。ディスクキャッシュのヒット率が低い場合には、ディスク装置とディスクキャッシュ間のデータ転送量が増え、RPSミスが頻繁に発生する。これは、多数のディスク装置が少数のデータ転送(ディスクキャッシュとディスク装置間)パスを共有しているために起こる。スループットを向上させるためには、ディスクキャッシュとディスク装置間のパス数を増やす必要があることを述べた。また、ディスクキャッシュのヒット率が高い場合にも、高スループットを

得るためにはディスクキャッシュと入出力チャネル間のパス数を増やす必要がある。したがって、高スループットを得るためにはディスクキャッシュ自体のデータ転送幅を広げる必要があることを指摘した。本論文では、マルチポート･ページメモリ：MPPMをディスクキャッシュとして階層的に用いる提案をしている。

3.2節では、MPPMをディスクキャッシュとして階層的に用いたディスク･サブシステム：DIMPの構成とキャッシュ動作について述べた。まず、MPPMの基本構成と動作について述べた。MPPMは複数の入出力ポートをもち、どのポートからも独立にブロック単位でデータの入出力が行えるメモリであることを述べた。その後、DIMPの構成を述べた。DIMPではMPPMをディスクキャッシュとして階層的に用いることにより、ディスク装置とディスクキャッシュ間のデータ転送パスを十分に大きくできる。したがって、ディスクキャッシュとディスク装置間パスはボトルネックとならない。また、ライトスルー方式とライトバック方式の2種のキャッシュデータの置き換え方式に関して、DIMPのキャッシュ動作を解説した。ここで扱ったキャッシュ動作は、キャッシュデータのコヒレンシを考慮しないキャッシュ動作である。

3.3節では、従来のキャッシュ付きディスク･サブシステムとDIMPの処理性能に関して述べた。評価対象とするディスク装置の性能と評価条件を示した。2章で述べた待ち行列網シミュレータCABを用いてシミュレーションした結果から、従来のディスク･サブシステムではディスク装置とディスクキャッシュ間のデータ転送パスがボトルネックであることが示された。また、DIMPでは、入出力チャネルのデータ転送幅よって制限される、あるいは、ディスク装置の処理性能によって制限されるまでの高いスループットが得られることが示された。また、待ち行列網シミュレータCABを用いることにより、新たにプログラムを作成することなしに、対話的にシミュレーション実験が行えたことを述べた。

第4章 DIMPの二層全接続型構成におけるキャッシュ動作と処理性能

本章では、マルチポート·ページメモリをディスクキャッシュとして階層的に用いたディスク·サブシステム:DIMPの二層全接続型構成について述べる。DIMPの二層全接続型構成では、キャッシュデータのコヒレンシを考慮したキャッシュ動作が必要となる。キャッシュデータのコヒレンシを考慮した場合の4種のキャッシュ動作について述べる。また、それらのキャッシュ動作における処理性能を示し、それらの特性を明らかにする。

以下では、4.1節でDIMPの二層全接続型構成について述べる。また、ここではキャッシュデータのコヒレンシ問題についても解説する。4.2節でDIMPの二層全接続型構成におけるキャッシュデータのコヒレンシを考慮した4つのキャッシュ動作について述べる。また、キャッシュメモリ内のデータ格納方式についても述べる。4.3節でDIMPの二層全接続型構成の4種のキャッシュ動作における処理性能を示す。これらの処理性能を比較し特性を明らかにする。4.4節で本章のまとめを述べる。

4.1節　DIMPの二層全接続型構成

図4.1に示した構成をDIMPの二層全接続型構成と呼ぶ。これは、前章の図3.7に示した構成に対して、上層のMPPMを下層のMPPMと同数の8個にしたものである。これは、前章の図3.8に示した128本の入出力ポートを持つMPPMを1個用いたDIMPの構成と等価と考えられる。このような構成にすることで、どのチャネルからも、すべてのディスク装置にアクセスできる。したがって、非常に大きな入出力チャネルの転送幅が得られる。

DIMPの二層全接続型構成では、入出力チャネルと接続される独立なMPPMが複数存在する。したがって、キャッシュデータのコヒレンシの問題がある。以下ではキャッシュデータのコヒレンシ問題について解説する。

[キャッシュデータのコヒレンシ問題]

図4.2は、下層のMPPM1とそれと接続される上層のMPPM2と上層のMPPM3とのデータ転送における誤ったデータアクセス例を表している。これを例にキャッシュデータのコヒレンシ問題を解説する。

上層のMPPM2と上層のMPPM3は下層のMPPM1と接続されており、上層のMPPM2と上層のMPPM3は下層のMPPM1中のデータのコピーをそれぞれ保持しているとする。I/Oチャネルの1つからMPPM2へこのコピーデータの書き込み要

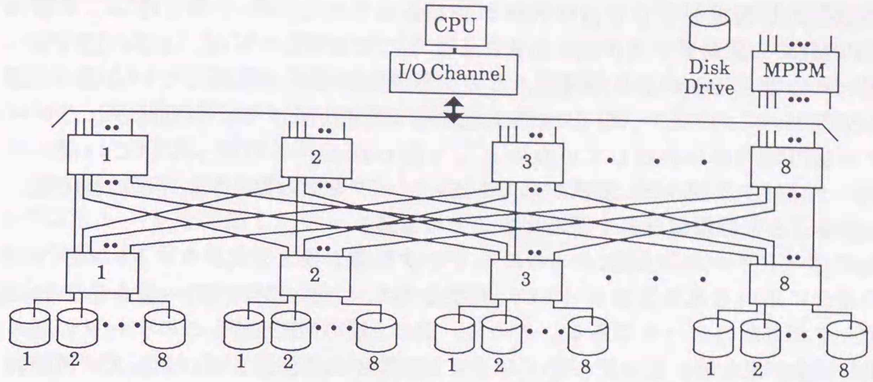


図 4.1　DIMPの二層全接続型構成

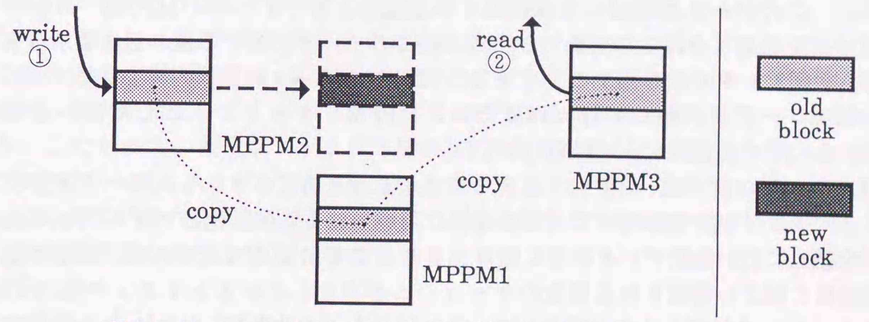


図4.2　キャッシュデータのコヒレンシ問題(誤ったデータアクセス例)

求が起き、その後、別のI/OチャネルからMPPM3へこのコピーデータの読み出し要求が起きたとする。最初のI/OチャネルからのMPPM2への書き込み要求でコピーデータは更新されている。この後、このデータの読み出しはこの更新されたデータを読み出さなければならない。にも関わらず、MPPM3からは古いデータが読み出されてしまう。これがキャッシュデータのコヒレンシ問題である。これはI/Oチャネルと独立に接続されるMPPMが複数存在するために起こる。キャッシュデータのコヒレンシ問題はディスクキャッシュばかりでなく、複数のCPUと独立に接続される複数の主記憶キャッシュをもつ場合(パラレルキャッシュ·システム[79])にも起こる。このような構成の場合には、正しくデータの読み出しが行われるよう工夫が必要となる。これらの工夫として無効化方式やブロードキャスト方式がある。また、パラレルキャッシュ·システムで用いられている特別な方式としてスヌーピング方式[80]がある。

図4.2に示された構成を例に、これらの方式を説明する。

1) 無効化方式:キャッシュデータの置き換え方式がライトスルー方式の場合に用いられる。ライトスルー方式では、上層のMPPM2へ書き込まれたデータは同時に下層のMPPM1へも書き込まれる。その後、別の上層のMPPM3へこのデータ

の読み出し要求が来ても古いデータが読み出されないよう無効化し、下層のMPPM1から最新データが読み出されるようにする方式である。上層のMPPMへデータの書き込みが生じた時に、このデータのコピーを保持している他の上層のMPPMへこのコピーデータを無効化するよう指示をする。無効化されたデータはMPPM内には存在していないとして扱われる。したがって、この後このデータに対する読み出し要求が来ても古いデータを読み出さず、下層のMPPMから読み込んだ最新データを読み出すことになる。

2) ブロードキャスト方式：キャッシュデータの置き換え方式がライトバック方式の場合に用いられる。ライトバック方式では、上層のMPPM2へ書き込まれたデータが最新のデータである。その後、別の上層のMPPM3へこのデータの読み出し要求が来た時、古いデータではなくMPPM2へ書き込まれた最新データが読み出されるようにする方式である。上層のMPPMへデータの書き込みが生じた時に、このデータのコピーを保持している他の上層のすべてのMPPMへ最新のデータを転送しそれらを更新する。この後このデータに対する読み込み要求が来ても最新データを読み出すことになる。MPPMのデータ転送は内部接続網に与える制御コードに同期して行われるため、MPPMをキャッシュとして用いる場合、この方式を実現するのは難しい。

3) スヌーピング方式：キャッシュデータの置き換え方式がライトスルー方式とライトバック方式を組み合せた特別な場合に用いられる。無効化とブロードキャストを適当に行い最新データが常に読み出されるようにする方式である。上層の処理装置と独立に接続される複数のキャッシュが存在しているとする。それらのキャッシュに対する入出力要求とデータ転送をすべて監視し、それらの動作状態により無効化とブロードキャストを行う。この方法では、すべてのキャッシュに対する入出力要求とデータ転送を監視するための特別な機構を設ける必要がある。

4.2節　DIMPの二層全接続型構成におけるキャッシュ動作

本節では、DIMPの二層全接続型構成に関して、キャッシュデータのコヒレンシを考慮した4種のキャッシュ動作について述べる。

キャッシュデータのコヒレンシを考慮した場合、ライトスルー方式では無効化方式が一般的な方法である。ライトバック方式ではブロードキャスト方式やスヌーピング方式が用いられる。したがって、ライトバック方式はライトスルー方式に比べて複雑な処理になる。MPPMでは、データのアクセスはスイッチングネットワークに与える制御コードに同期して行われる。したがつて、MPPMをキャッシュとして用いる場合には、無効化はキャッシュデータを管理しているディレクトリのマッピング･テーブル中のエントリを削除することで容易に行えるが、ブロードキャストやスヌーピングを用いる方式は難しい。そこで、下

層のMPPMが、それと接続されている上層のMPPMに対して、強制的に書き戻し処理が行えるようにし、無効化方式と組み合せて用いる。下層のMPPMにリード要求が起きた時、上層のMPPMが要求ブロックの最新ブロックを持っている場合には、強制的に該当ブロックを書き戻させ、それを出力することで、キャッシュデータの首尾一貫性が保てる。

先の図4.2に示された例において、ライトスルー方式とライトバック方式の場合のコヒレンシを考慮したデータ転送について、以下で簡単に説明する。図4.3および図4.4に示されるのが、それぞれライトスルー方式とライトバック方式の場合のキャッシュデータのコヒレンシを考慮したデータ転送である。

(1)ライトスルー方式

MPPM2に書き込み(図中①)が生じたとき、この書き込みブロックのコピーをもつ他の上層のMPPM3には無効化信号が送られる(図中Invalidate)。この無効化信号により、この該当ブロックは無効化されMPPM3には存在しないブロックとなる。さらに、MPPM2に書き込まれた最新ブロックは対応する下層のMPPM1に書き込まれる(図中②)。この後、MPPM3に対して、このブロックに対する読み出し要求が発生してもこのブロックはMPPM3には存在せずリードミスとなる。したがって、要求ブロックは下層のMPPM1から読み出され(図中③)、それがMPPM3へ読み込まれ、さらに上層へ転送される(図中④)。MPPM3に対する読み出し要求が図中②の処理終了前に生じた場合には、この処理が完了するまで、以後の処理(図中③および④)は待たされる。このようにしてデータの首尾一貫性が保たれる。

(2)ライトバック方式

MPPM2に書き込み(図中①)が生じたとき、この書き込みブロックのコピーをもつ他の上層のMPPM3には無効化信号が送られる(図中Invalidate)。この無効化信号により、この該当ブロックは無効化されMPPM3には存在しないブロックとなる。この後、MPPM3に対して、このブロックに対する読み出し要求が発生してもこのブロックはMPPM3には存在せずリードミスとなる。対応する下層のMPPM1に対して読み込み要求が発生する。MPPM1は、上層の他のMPPMに対

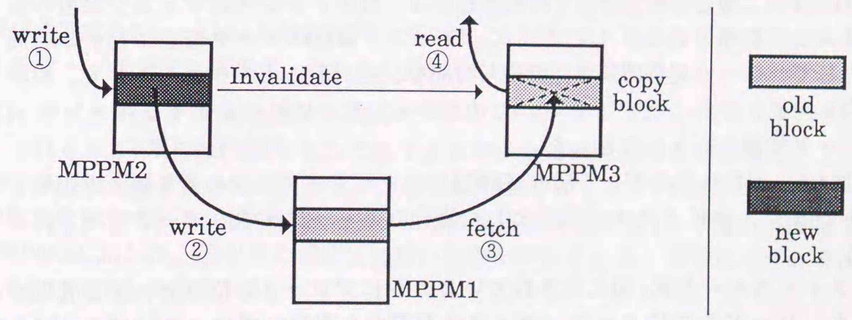


図4.3　キャッシュデータのコヒレンシを考慮したキャッシュのデータ転送(ライトスルー方式)

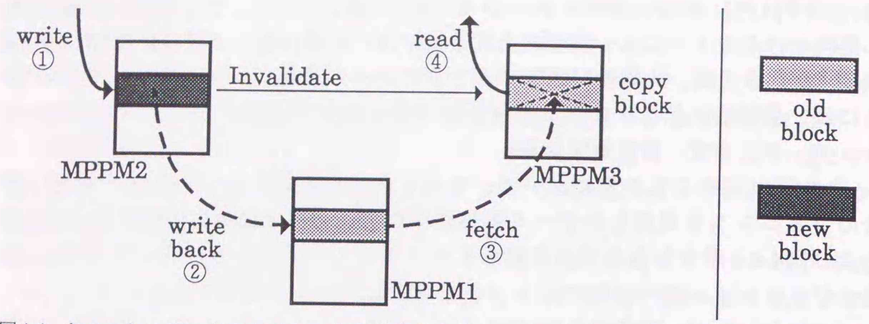


図4.4 キャッシュデータのコヒレンシを考慮したキャッシュのデータ転送(ライトバック方式)

して、該当ブロックの最新ブロックをもつかをチェックし、最新ブロックをもつものがあればそれを書き戻させる(図中②)。このブロックを要求のあったMPPM3に対して転送し(図中③)、それがMPPM3から上層に転送される(図中④)。このようにして、データの首尾一貫性が保たれる。

上記は、二層に構成されているMPPMのうち、上層のMPPMのキャッシュ動作がそれぞれライトスルー方式およびライトバック方式の場合を表している。キャッシュデータの置き換え方式としては、上/下層のMPPMについてライトスルーとライトバック方式をそれぞれ採用可能であるから、以下の4方式が考えられる。

[1] ライトバック/ライトバック(以下WB/WBと略記する)
[2] ライトスルー/ライトスルー(以下WT/WTと略記する)
[3] ライトスルー/ライトバック(以下WT/WBと略記する)
[4] ライトバック/ライトスルー(以下WB/WTと略記する)

以降では、これらのキャッシュ動作について詳しく説明する。各キャッシュ動作におけるデータ転送を理解し易いよう、キャッシュ中のデータ管理について先に述べる。

[キャッシュ中のデータ管理]

DIMPの二層全接続型構成での限界性能は、後述するようにディスク装置の台数によって制限される。したがって、ディスク装置のデータ転送に要する時間に対して、シーク動作時間と回転待ち時間を、相対的に小さくすることで、限界性能を向上させることができる。この点から、以下に述べるようなキャッシュのデータ管理を行うものとした。

図4.5に示されるのが、下層のMPPMのキャッシュデータの置き換え方式をライトバック、ライトスルー方式とした場合のキャッシュ中のデータ管理方式である。

1) ライトスルー方式:図に示されているようにブロック単位でデータを管理する。リードミスによって、ディスク装置から複数のブロック(あるいは1トラック·データ)を読み込む場合、それらのブロックは、LRU規則により決められたブロックと置き換えられ格納される(図中①)。置換されたブロックが、更新されたブロックであっても、これらは、更新時にすでにディスク装置に書き込まれているので捨ててしまってよい(図中②③)。

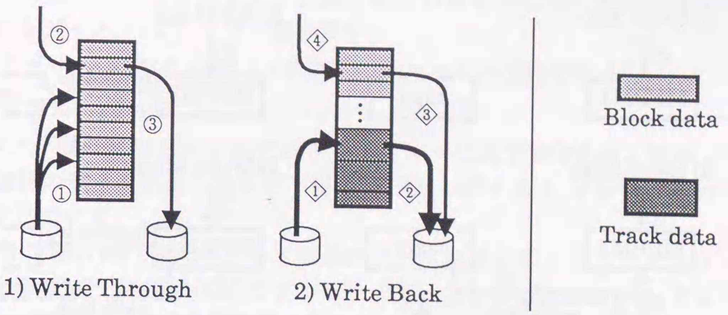


図4.5 キャッシュのデータ管理

2) ライトバック方式:リードミスによって、ディスク装置から複数のブロック(あるいは1トラック·データ)を読み込む時(図中①)、LRU規則によって置換ブロックが選ばれる。これらが更新されている場合には、ディスク装置へ書き戻す必要がある。この時、ブロック単位のデータ管理では、置換ブロックは複数存在し、複数のディスク装置へ書き戻し動作を行わなければならない。したがって、図に示されているように、ブロック単位のデータ領域とトラック単位のデータ領域に分け、領域の境界は動的に変化させるようにする。ディスク装置からトラック·データを読み込む時には、置換ブロックは、トラック単位のデータ領域から選ばれる。これが更新されたブロックならば、対応する1つのディスク装置にだけ書き戻しの動作を行えばよい(図中②)。同様に、ライトミスで上位装置からキャッシュにデータを書き込む時も(図中④)、置換ブロックは、ブロック単位のデータ領域から選ばれる。これが更新されたブロックであっても対応する1つのディスク装置にだけ書き戻しの動作を行えばよい(図中③)。

同様の方式をNarashimaらは提案[81]している。以下では、各キャッシュ動作におけるデータ転送図を挙げて、それらの動作をおのおの解説する。

[1] ライトバック/ライトバック方式

図4.6は上下層のMPPMをそれぞれライトバック方式で動作させた時の、上下層のMPPM間のデータ転送、および下層のMPPMとディスク·ドライブ間のデータ転送を示している。図中において、上層のMPPMをMPPM1、下層のMPPMをMPPM2と記した。①②③の添字は処理の順番を示している。以下に示される図4.7,4.8,4.9でも同様である。

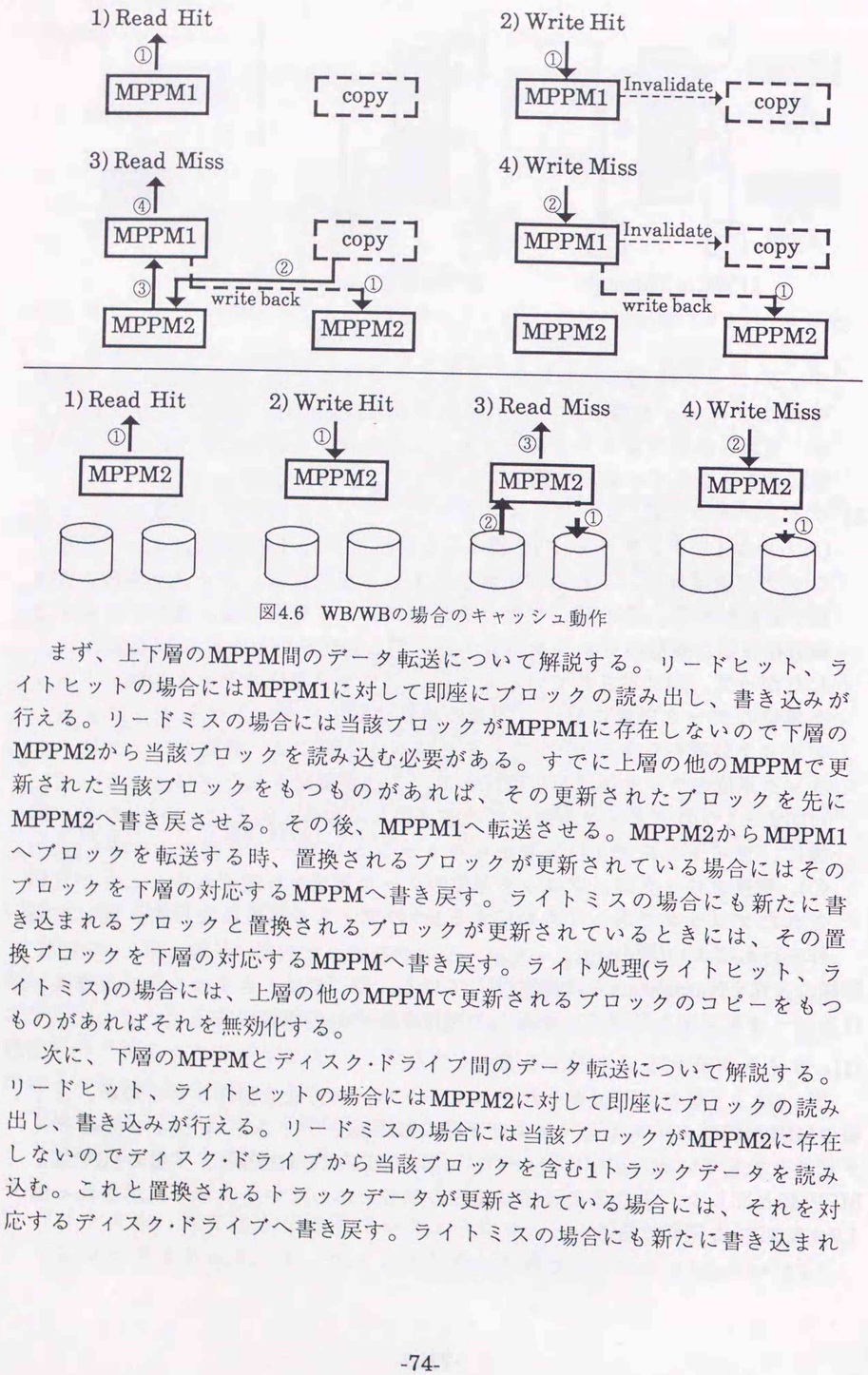


図4.6　WB/WBの場合のキャッシュ動作

まず、上下層のMPPM間のデータ転送について解説する。リードヒット、ライトヒットの場合にはMPPM1に対して即座にブロックの読み出し、書き込みが行える。リードミスの場合には当該ブロックがMPPM1に存在しないので下層のMPPM2から当該ブロックを読み込む必要がある。すでに上層の他のMPPMで更新された当該ブロックをもつものがあれば、その更新されたブロックを先にMPPM2へ書き戻させる。その後、MPPM1へ転送させる。MPPM2からMPPM1へブロックを転送する時、置換されるブロックが更新されている場合にはそのブロックを下層の対応するMPPMへ書き戻す。ライトミスの場合にも新たに書き込まれるブロックと置換されるブロックが更新されているときには、その置換ブロックを下層の対応するMPPMへ書き戻す。ライト処理(ライトヒット、ライトミス)の場合には、上層の他のMPPMで更新されるブロックのコピーをもつものがあればそれを無効化する。

次に、下層のMPPMとディスク･ドライブ間のデータ転送について解説する。リードヒット、ライトヒットの場合にはMPPM2に対して即座にブロックの読み出し、書き込みが行える。リードミスの場合には当該ブロックがMPPM2に存在しないのでディスク･ドライブから当該ブロックを含む1トラックデータを読み込む。これと置換されるトラックデータが更新されている場合には、それを対応するディスク･ドライブへ書き戻す。ライトミスの場合にも新たに書き込まれ

るブロックと置換されるブロックが更新されているときには、その置換ブロックを対応するディスク･ドライブへ書き戻す。

以上がライトバック/ライトバック方式のキャッシュ動作である。

[2]　ライトスルー/ライトスルー

図4.7は上下層のMPPMをそれぞれライトスルー方式で動作させた時の、上下層のMPPM間のデータ転送、および下層のMPPMとディスク･ドライブ間のデータ転送を示している。

まず、上下層のMPPM間のデータ転送について解説する。リードヒットの場合には、上層のMPPM1から即座にブロックの読み出しが行える。リードミスの場合には、当該ブロックがMPPM1に存在しないので下層のMPPM2から読み込む必要がある。この時、置換されるブロックが更新されていてもすでに下層のMPPMへ書き込まれているので捨ててしまってよい。その後、MPPM1から当該ブロックが読み出される。ライトヒットとライトミスの場合には、一旦更新ブロックを上層のMPPM1へ書き込む。置換されるブロックが更新されていても捨ててしまってよい。その後、この更新ブロックは下層のMPPM2へ書き込まれる。最終的にはディスク･ドライブへ書き込まれる。上層のMPPM1への書き込み

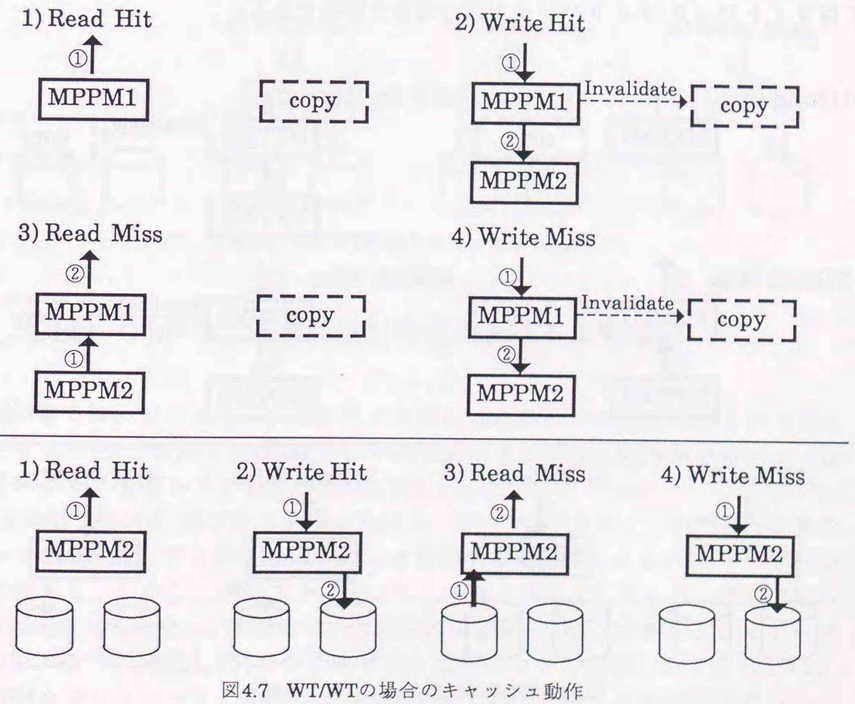


図4.7　WT/WTの場合のキャッシュ動作

時、上層の他のMPPMで更新されるブロックのコピーをもつものがあればそれらを無効化する。

次に、下層のMPPMとディスク･ドライブ間のデータ転送について解説する。リードヒットの場合には、MPPM2から即座にブロックの読み出しが行える。リードミスの場合には、当該ブロックがMPPM2に存在しないのでディスク･ドライブから当該ブロックを含む1トラックデータを読み込む。これと置換されるトラックデータが更新されていても、すでに更新データは対応するディスク･ドライブへ書き込まれているので捨ててしまってよい。ライトヒットとライトミスの場合には、一旦更新ブロックを下層のMPPM2へ書き込む。置換されるブロックが更新されていても捨ててしまってよい。その後、この更新ブロックはディスク･ドライブへ書き込まれる。

以上がライトスルー/ライトスルー方式の場合のキャッシュ動作である。

[3] ライトスルー/ライトバック

図4.8は上下層のMPPMをそれぞれライトスルー、ライトバック方式で動作させた時の、上下層のMPPM間のデータ転送、および下層のMPPMとディスク･ドライブ間のデータ転送を示している。

上下層のMPPM間のデータ転送については、ライトスルー/ライトスルー方式の場合と同様である。下層のMPPMとディスク･ドライブ間のデータ転送についてはライトバック/ライトバック方式の場合と同様である。

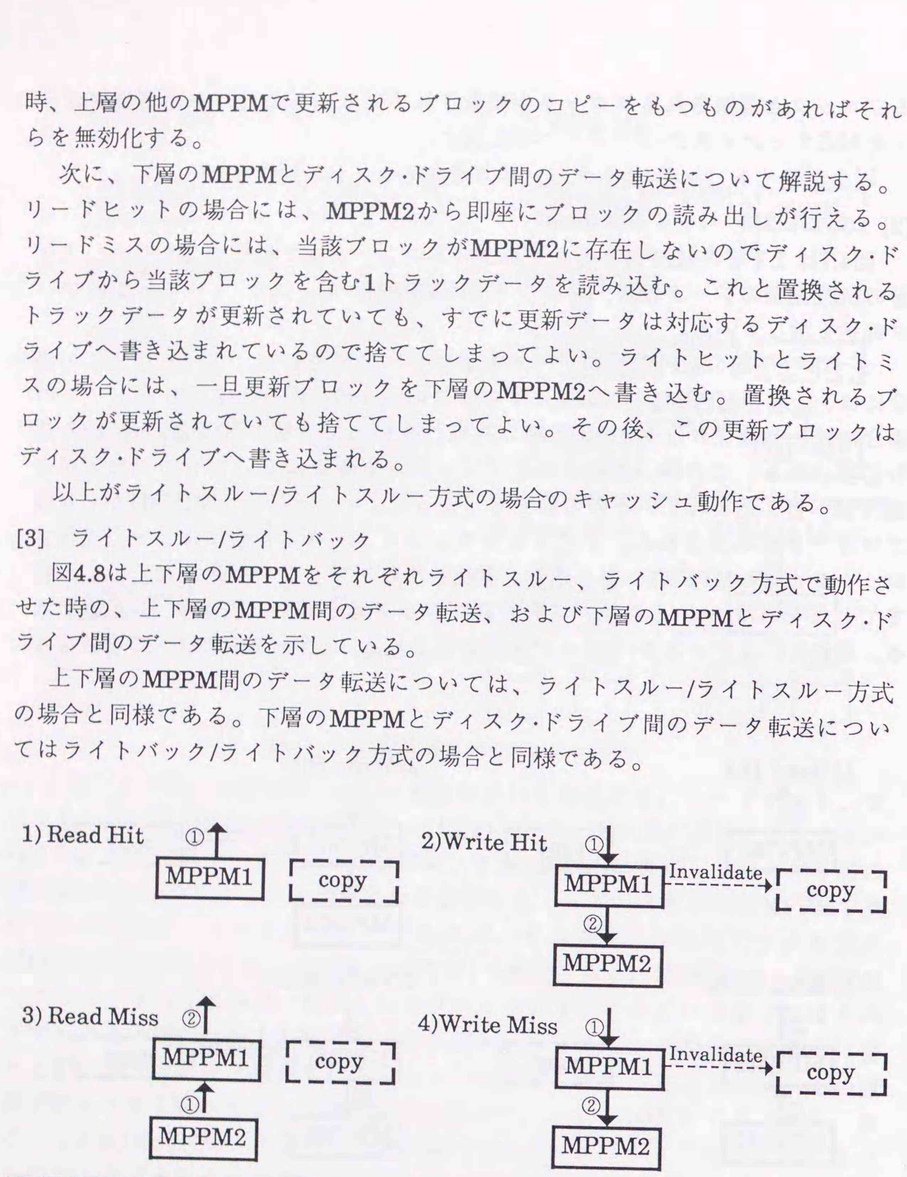


図4.8 WT/WBの場合のキャッシュ動作

[4] ライトバック/ライトスルー

図4.9は上下層のMPPMをそれぞれライトバック、ライトスルー方式で動作させた時の、上下層のMPPM間のデータ転送、および下層のMPPMとディスク･ドライブ間のデータ転送を示している。

上下層のMPPM間のデータ転送については、ライトバック/ライトバック方式の場合と同様である。下層のMPPMとディスク･ドライブ間のデータ転送についてはライトスルー/ライトスルー方式の場合と同様である。

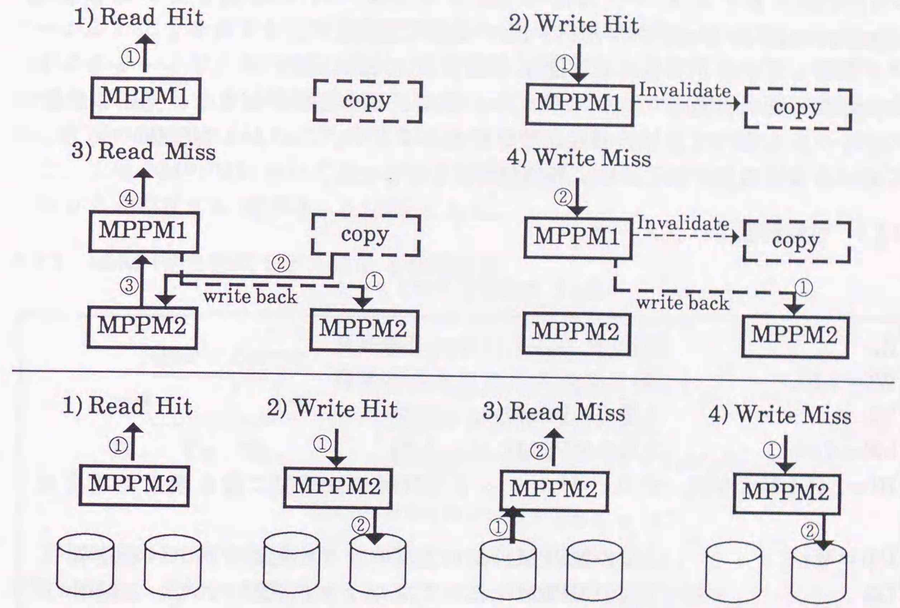


図4.9 WB/WTの場合のキャッシュ動作

4.3節 DIMPの二層全接続型構成における処理性能

前3章では、シミュレーションにより得られた結果から図3.7に示される構成のディスク･サブシステムDIMPと、それと対応する構成のMPPMを用いない従来型ディスク･サブシステムとの性能比較を行った。

従来型ディスク･サブシステムの中には、ディスク･ドライブ内部に小容量のバッファを設け、ディスクキャッシュと併用する階層型キャッシュ[64][65]を持つものがある。しかし、ディスクキャッシュ自体を階層化したものは著者の知る限り提案されていない。前節では、二層全接続型構成に関してキャッシュデータのコヒレンシを考慮した4つのキャッシュ動作について説明した。これらは、MPPMをキャッシュとして用いる場合に有効であると考えられる動作である。

これら4つの動作に限定して、解析的な手法により性能評価を行った。本節では、これらの評価結果を示し4種のキャッシュ動作におけるDIMPの処理性能を考察し、その特性を明らかにする。

前章の性能評価と同様に、待ち行列網シミュレータCABを用いて評価実験を行うことにより、より現実に近い評価結果が得られる。ただ、本章の対象システム(図4.1に示される構成)は、前章の対象システム(図3.8に示される構成)よりもキャッシュ動作が複雑であり、多くの基本モデルを用いる必要がある。これに伴い、シミュレーションの実行時間とシミュレータが稼働している計算機(Symbolics 3620:Lisp言語専用計算機)の必要記憶容量が大きくなる。シミュレータが稼働している計算機の性能(特に記憶容量)に制限され、シミュレーションによる性能評価が現実的ではなくなった。また、この性能評価では、先述した4つのキャッシュ動作における性能比較が行えればよい。このような理由から、以下に述べるような解析的手法により性能評価を行った。

4.3.1 性能評価用パラメータ

表4.1 性能評価用パラメータ

R.	I/Oリクエストのリードの割合
W(=1-R).	I/Oリクエストのライトの割合
h1.	上層のMPPMのヒット率
h2(=4×h1).	下層のMPPMのヒット率
H(=1-(1-h1)(1-h2)).	グローバルヒット率(1個のMPPMに置き換えた場合のヒット率)
D1(=W).	上層のMPPMにおいてブロックが更新されている確率
D2.	下層のMPPMにおいてブロックが更新されている確率
S(=0.1に固定).	上層のMPPMにおいて同一ブロックを共有している確率
Tb.	1ブロックの転送時間
Ttr.	1トラックの転送時間
Tsw.	シークと回転待ちの平均時間
Np.	MPPMの入出力ポート数
Tbyte.	1バイト(1ポート)の転送時間
Ts(=Np×Tbyte).	1スライスの転送時間
Tpw(=1/2×Ts).	ポートの切り替えによる待ち時間
Cp=64.	接続される全チャネル数
Ip=64.	上下層のMPPM間で接続されるパス数
D=64.	全ディスク・ドライブ数
Th.	スループット

待ち行列網解析(M/M/1待ち行列モデル)[71]により性能評価を行った。表4.1に使用したパラメータの一覧を示す。評価対象とするディスク装置の性能は前章の表3.1と同様である。評価条件は前章の表3.2中(3)~(5)に示されたものと同様である。

下層のMPPMのヒット率は、上層のMPPMのヒット率の4倍と仮定した。以下の文中、ヒット率とは、グローバルヒット率を表すものとする。これは、上下層のMPPMのヒット率を、1つのキャッシュメモリのヒット率に置き換えたものである。すなわち、上層のMPPMのキャッシュヒット率がh1、下層のMPPMのキャッシュヒット率がh2とすると、グローバルヒット率はH=(1-(1-h1)×(1-h2))で表される。上層のMPPMにおいてブロックが更新されている確率は、ライトの割合と同一とする。下層のMPPMにおいてブロックが更新されている確率は、下層のMPPMに対するアクセスにおけるライトの割合と同一であると仮定した。上層のMPPMにおいて同一ブロックを共有している(他のMPPMにコピーブロックが存在する)確率は、0.1固定とした。

4.3.2 M/M/1待ち行列モデルによる性能評価

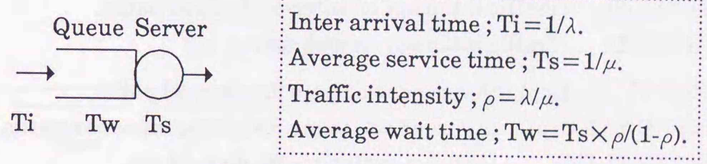
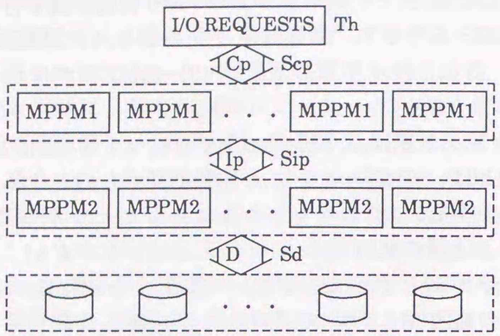


図4.10 M/M1待ち行列モデル

評価対象のDIMPの構成は、先の図4.1に示したものである。ディスク・ドライブ数が64台、チャネル数が64である。システムの平均応答時間は、システムの各パート(チャネル、上下層のMPPM間で接続されるパス、ディスク・ドライブ)での平均処理(サービス)時間と、各パートでの平均待ち時間をすべて加えることにより求まる。システムが定常状態にあるとき、図4.10に示すM/M/1待ち行列モデルを用いることにより、トラフィック密度とサーバでの平均サービス時間から待ち行列での平均待ち時間が求まる[71]。ただし、各パートでのジョブの到着時間とサービス時間は他のパートと独立で、それぞれポアソン過程に従うものと仮定する。また、特定のディスク・ドライブ装置へ集中的に入出力要求が発生するアクセス集中は生じないと仮定する。したがって、上記の各パートを個々にM/M/1待ち行列モデルに置き換えることができる。先の図4.1に示した構成のシステムにM/M/1待ち行列モデルを適用すると、各パートでの平均待ち時間が図4.11に示されるように計算される。各パートでの平均処理(サービス)時間は、表4.1のパラメータより計算される。4つのキャッシュ動作方式における具体的な計算式は付録Iに示した。



Th ;Throughput

Scp ;Average service time on channel paths.
Sip ;Average service time on inter-connection paths.
Sd ;Average service time on disk drives.

Ucp=Th×Scp/Cp. ;Traffic intensity on channel paths.
Uip=Th×Sip/Ip. ;Traffic intensity on inter-connection paths.
Ud =Th×Sd /D. ;Traffic intensity on disk drives.

Wcp=Scp×Ucp/(1-Ucp). ;Average wait time for channel paths.
Wip=Sip×Uip/(1-Uip). ;Average wait time for inter-connection paths.
Wd =Sd×Ud/(1-Ud). ;Average wait time for disk drives.

図4.11 DIMPの二層全接続型構成へ適用したM/M/1待ち行列モデル

この評価ではM/M/1待ち行列モデルを適用したが、より現実的な評価ではM/G/1待ち行列モデル[71]が適用される。M/G/1待ち行列モデルでは、平均待ち時間は以下の式で表される。

Average wait time : $Tw=(1+C^2)/2\times Ts\times \rho/(1-\rho)$.

ただし、$C^2=Var[s]/(E[s]^2)$. Var[s]とE[s]はサービス時間の分散と平均である。C^2はサービス時間sの"平方変動係数"と呼ばれる係数である。この式は、M/M/1待ち行列モデルでの平均待ち時間に係数$(1+C^2)/2$を掛けたものである。すなわち、M/M/1待ち行列モデルでは$C^2=1$である。また、M/D/1待ち行列モデル[71]では$C^2=0$である。$C^2>1$の場合には、M/M/1待ち行列モデルの場合よりもその平均待ち時間が大きくなる。このよに、M/G/1待ち行列モデルを適用する場合には、サービス時間の分布特性(分散)が明らかでなければならない。本評価では、対象システムの動作が複雑であり、それらを求めることが難しい。したがって、上記のようにM/M/1待ち行列モデルを適用し性能評価を行った。ただ、M/D/1待ち行列モデル($C^2=0$)を適用した場合や$C^2=10$程度にC^2の値が大きな場合でも、以下に述べるのと同様の比較結果になることを確認している。

4.3.3 評価結果

リードの割合とキャッシュのヒット率を変化させ、各キャッシュ動作(WB/WB方式、WT/WB方式、WB/WT方式、WT/WT方式)について、スループットと応答時間を求め比較した。

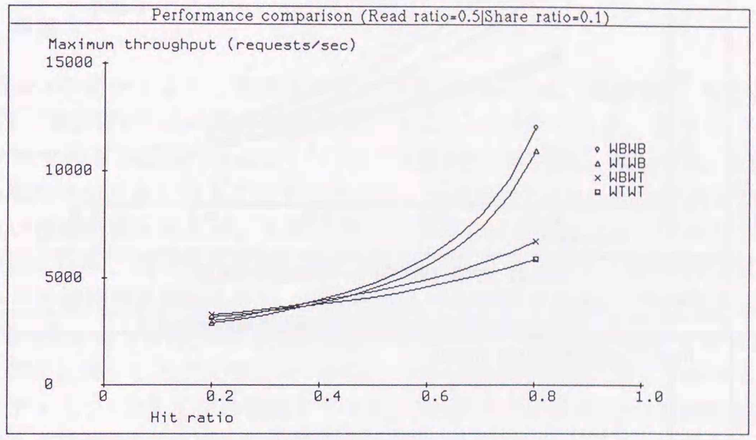


図4.12 性能比較(1)(リードの割合=0.5)

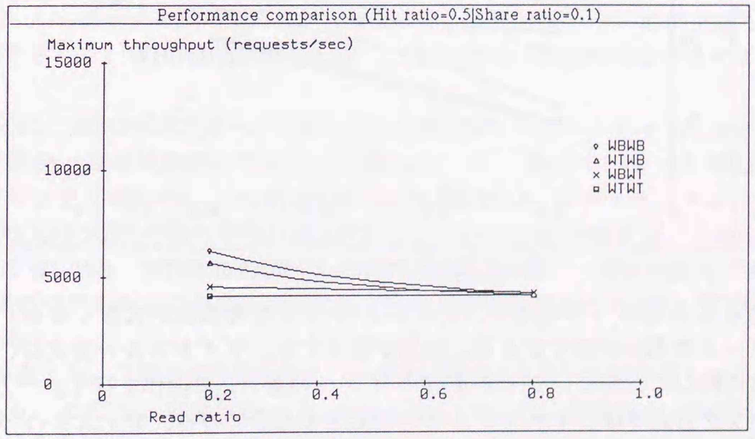


図4.13 性能比較(2)(ヒット率=0.5)

図4.12は、リードの割合を0.5に固定した場合の、ヒット率に対する限界スループットを表したグラフである。これは、ディスク・ドライブの利用率が100%になる時のスループットを示している。ヒット率が上昇すると、ディスク・ドライブへのアクセスが減り、性能が向上する。図4.13は、ヒット率を0.5に固定した場合の、リードの割合に対する限界スループットを表したグラフであ

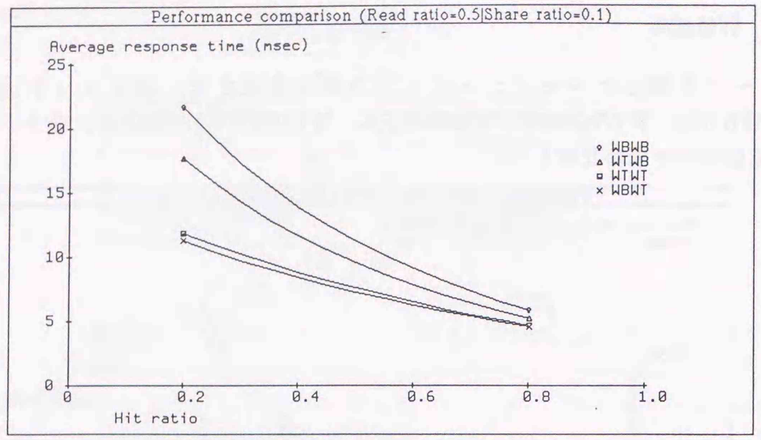


図4.14　性能比較(3)(リードの割合＝0.5)

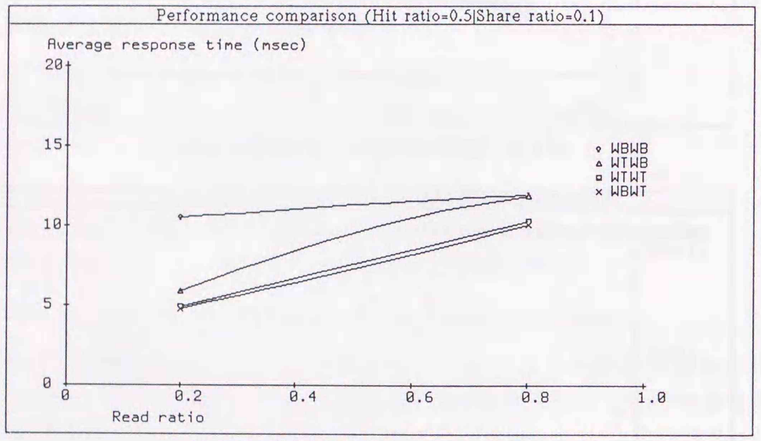


図4.15　性能比較(4)(ヒット率＝0.5)

る。ライトよりリードの方が、ディスク･ドライブの負荷が大きくなる。よって、リードの割合が高くなると、性能が低下する。ライトスルー方式は、ライト要求に対して、常に、ディスク･ドライブへの書き込み動作を行う。一方、ライトバック方式では、キャッシュ･ミスが起きた場合にのみ、ディスク･ドライブへアクセスする。したがって、ヒット率が高い程、あるいはライトの割合が高い程、ライトバック方式がライトスルー方式に比べて性能が良くなる。

図4.14は、スループットが1,000(IO要求/秒)で、リードの割合が0.5の場合の、ヒット率に対する平均の応答時間を表したグラフである。ヒット率が上昇すると、ディスク･ドライブへのアクセスが減り、性能が向上しているのが分かる。図4.15は、スループットが1,000(IO要求/秒)で、ヒット率が0.5の場合の、リードの割合に対する平均の応答時間を表したグラフである。図4.13のグラフと

同様に、リードの割合が高くなると、性能が低下しているのが分かる。ライトスルー方式は、ライト要求に対して、キャッシュのヒット/ミスに関係なく、常に書き込み動作が行える。よって、ライトバック方式に比べて、ライトスルー方式の方が、平均の応答時間は良くなる。ヒット率が低い程、あるいはライトの割合が高い程、その差が大きくなるのが分かる。

4.3.4　考察

上記4つのグラフより、限界スループットについては、WB/WB、WT/WB、WB/WT、WT/WT方式の順に性能が良くなることが分かった。低トランザクション域での平均応答時間については、WB/WT、WT/WT、WT/WB、WB/WB方式の順に性能が良くなることが分かった。WB/WT方式は、応答時間については、良い性能が得られるが、スループットは非常に悪い。また、キャッシュ動作が複雑になり、実現性の点から不利である。これに比べて、WT/WB方式は、キャッシュ動作が比較的単純で、実現性の点から有利である。WT/WB方式は、WB/WB方式とWT/WT方式の中間的な性能が得られ、WB/WB方式およびWT/WT方式の欠点を補うことができ、4つのキャッシュ動作中最適な方式であると思われる。ディスク･ドライブの性能を先の表3.1に示される値の2分の1(回転速度を2倍、シーク時間を2分の1)とした場合や、転送ブロックサイズを1トラック･データの半分程度にした場合でも同様の結果が得られた。ただ、転送ブロックサイズが1トラック･データと同程度に大きな場合には、応答時間については上記と異なる傾向を示した。WB/WB方式が最も悪く、他の3者は同程度の性能を示すようになる。

さらに、128の入出力ポートを持つ1個のMPPMをディスクキャッシュとして用いた場合(一層単純構成と呼ぶ)との比較も行った。表4.2は、ヒット率0.65(グローバルヒット率)、リードの割合0.8の場合の限界スループットと、スループットが100(IO要求/秒)の時の平均応答時間を示している。上段2項目は、二層全接続型構成のWB/WB、WT/WT方式の値である。3、4段目は、一層単純構成のWB、WT方式の値である。この場合の動作は、先の3.3節で説明されたWB、WT方式と同じである。

この表より、二層全接続型構成は、一層単純構成より良い性能が得られることが分かる。ただし、命令転送(リード/ライトコマンド、無効化など)のオーバヘッドは含まれていない。二層全接続型構成は、二層に構成することで、データ転送のオーバヘッドが増すが、少数の入出力ポートを持つMPPMを採用することで、ポート切り替えによる待ち時間が小さくなる。表4.2の結果は、このオーバヘッドが打ち消し得ることを示している。また、表4.2の5、6段目は、二層全接続型構成で、下層のMPPMのヒット率を0.65とし、上層のMPPMのヒット率をその4分の1とした場合の性能である。下層のMPPMのみのヒット率が、一層単純構

表4.2　性能比較(5)(リードの割合＝0.8)

	限界スループット (requests/sec)	平均応答時間 (msec)
WB/WB①	6,060	7.5
WT/WT①	5,450	6.6
WB	6,040	7.5
WT	5,440	6.7
WB/WB②	7,240	6.7
WT/WT②	6,190	6.0
BD	1,410	6.76

成の単一MPPMのヒット率と同じ場合、二層全接続型構成では、上層のMPPMのキャッシュ効果により、一層単純構成よりもさらに性能が良くなる。

また、MPPMをキャッシュとして用いない従来のディスク･サブシステムの性能を、参考データとして表4.2の7段目に示した。チャネル･パス数が8、ディスク･ドライブとディスクキャッシュ間パス数が8、ディスク･ドライブ台数が64で、バッファ内蔵型ディスク･ドライブ装置を用いたディスク･サブシステムである。この値は、シミュレーションにより得られた値である。

以上で示した限界スループットは、ディスク装置の台数とその性能がボトルネックとなっている。ディスク･ドライブに専用に接続されるパスを、そのディスク･ドライブがアクセスされてない時や、シーク待ちや回転待ちでデータ転送が行われてない時に、他のディスク･ドライブのデータ転送に利用できるようなダイナミック･クロスパス構成にすることで、ディスク台数を増やすことができ、限界スループットをさらに向上できる。ただし、この場合にはRPSミスによる回転待ちが頻繁に生じるため、ディスク･ドライブ内蔵型バッファを用いてディスク･ドライブの回転と非同期的にデータ転送が行える必要がある。

4.4節　4章の要約

本章では、MPPMをキャッシュメモリとして階層的に用いたディスク･サブシステムDIMPについて、特に、上下層に同数のMPPMを配した二層全接続型構成に関して述べた。DIMPの二層全接続型構成では、キャッシュデータのコヒレンシ問題がある。キャッシュデータのコヒレンシを考慮した4種のキャッシュ動作とそれらの処理性能を示し、その特性を明らかにした。

4.1節で、DIMPの二層全接続型構成について述べた。ここで、キャッシュデータのコヒレンシ問題についても簡単に説明した。二層全接続型構成では、下層のMPPMはディスク装置と接続され、上層のMPPMは入出力チャネルと接続される。この構成では、入出力チャネルと接続される独立なMPPMが複数ある。したがって、キャッシュデータのコヒレンシ問題がある。コヒレンシとは、異なる上位装置から独立にデータの読み書きが行われる場合に、他の上位装置が書き込んだデータを、他の上位装置が読み込む場合に常に最新の(最も近い時間で更新された)データを読み出さなければならないというデータの整合性をいう。

4.2節で、DIMPの二層全接続型構成における、キャッシュデータのコヒレンシを考慮した4つのキャッシュ動作を示した。キャッシュデータの置き換え方式としてライトバック(WB)方式とライトスルー(WT)方式が考えられる。それぞれについてキャッシュデータのコヒレンシを考慮したキャッシュ動作を示した。ライトスルー方式では無効化方式を採用した。ライトバック方式では無効化方式と強制書き戻し処理を採用した。また、上下層のMPPMそれぞれについて、ライトスルー方式とライトバック方式が考えられる(WB/WB方式、WT/WB方式、WB/WT方式、WT/WT方式)。これら4種のキャッシュ動作を限定し、それぞれの場合のキャッシュ動作を示した。

4.3節で、上記4種のキャッシュ動作における性能評価を行い、これらキャッシュ動作の特性を明らかにした。M/M/1待ち行列モデルを適用した解析的評価により得られた処理性能から、限界スループットおよび平均応答時間を比較し以下の結果を得た。

･限界スループット

WB/WB＞WT/WB＞WB/WT＞WT/WT　(良い＞悪い)

･(低トランザクション域での)平均応答時間

WB/WT＞WT/WT＞WT/WB＞WB/WB　(良い＞悪い)

本章では、これらの結果と実現の容易さから、WT/WB方式が最適な方式であることを述べた。

第5章 DIMPのディスクアレイ·システムにおけるキャッシュ動作と処理性能

前3,4章では、マルチポート·ページメモリをディスクキャッシュとして階層的に用いたディスク·サブシステム：DIMPの構成とキャッシュ動作および処理性能について述べた。従来のキャッシュ付きディスク·サブシステムでは、ディスクキャッシュとディスク装置間のデータ転送がボトルネックとなっていることを示した。DIMPではMPPMをディスクキャッシュとして用いることにより、小さなオーバヘッドでディスクキャッシュとディスク装置間のデータ転送幅を増やすことができ高スループットが得られることを示した。

本章では、ディスク装置の信頼性(耐故障性)を上げる手法であるディスクアレイ·システムに関して述べる。複数台のディスク装置を組みにして用い、各組みにそれぞれ少数台の冗長ディスクを付加することにより信頼性を上げるというのがディスクアレイ[44]である。MPPMをディスクキャッシュとして用いた場合のディスクアレイ·システムにおけるキャッシュ動作と処理性能を示す。ディスクアレイではチェックディスクに対する冗長なアクセスが必要となり、チェックディスクを用いない場合に比べてスループットが低下する。だが、MPPMをディスクキャッシュとして用いたディスクアレイ·システムでは、既存のキャッシュ付きディスク·サブシステムよりも優れた性能が得られる。

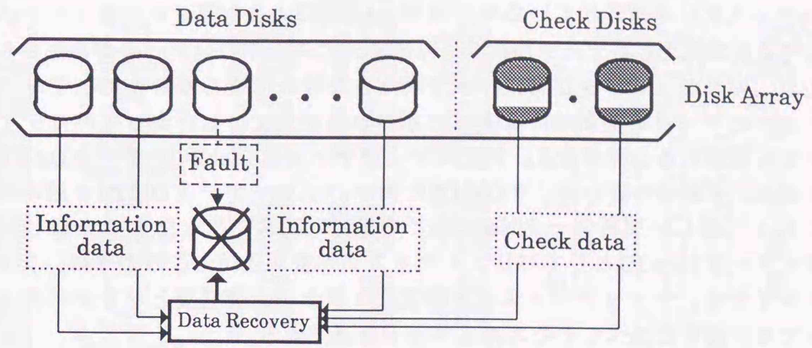
以下では、5.1節でまずディスクアレイ·システムについて述べる。5.2節でDIMPのディスクアレイ·システムにおけるキャッシュ動作について述べる。5.3節でDIMPのディスクアレイ·システムにおける処理性能について述べる。ディスクアレイ·システムの場合のDIMPの処理性能と既存のディスク·サブシステムの処理性能を挙げて比較する。5.4節で本章のまとめを述べる。

5.1節　DIMPのディスクアレイ·システム

本節ではディスクアレイ·システムについて解説する。

5.1.1　ディスクアレイ·システム

ディスクアレイとは、複数台のデータディスクを組みにして用い、それらの組みに対して少数台のチェックディスクを付加することにより信頼性をあげるものである。データディスクとはユーザデータが格納されるディスク装置である。チェックディスクとはチェックデータが格納されるディスク装置である。データディスクとチェックディスクを合わせてディスクアレイという。ディス



4 Data Disks (D1 D2 D3 D4) & Parity Disk (P1)

P1 = D1 XOR D2 XOR D3 XOR D4 (1)

Disk3 fault & data D3 recovery　　D3 = D1 XOR D2 XOR D4 XOR P1 (2)

Update data D2 on Disk2　New P1 = Old P1 XOR Old D2 XOR New D2 (3)

図5.1　ディスクアレイの故障ディスク回復機構

クアレイ中の任意の1台のディスク装置が故障した時に、その故障ディスクに格納されていたデータを、他の故障していないディスク装置に格納されているデータから回復できるようなデータをチェックデータとしてチェックディスクへ格納する。図5.1に示されるように、ディスクアレイ中の任意の1台のディスク装置が故障した場合には、他のデータディスクとチェックディスクに格納されているデータから故障ディスクのデータを回復可能となる。ただし、ディスクアレイ中の1台のディスク装置が故障中に他の1台のディスク装置が故障した場合には、それら故障ディスクのデータは回復不能となる。現実には2台のディスク装置が同時に故障する確率は非常に低いので信頼性が向上する。1エラー検出·訂正が行えるハミングコードや、1エラー検出あるいは1エラー訂正が行えるパリティなどがチェックデータとして用いられる。チェックデータの形式とデータのディスク装置への格納形式によりいくつかの方式が考えられる。図5.1下部に示されているのは、4台のデータディスクと1台のチェックディスク(パリティディスクとも呼ぶ)を用いた場合のディスク故障回復時とデータの更新時の計算式である。4台のデータディスクの同一位置には、4つのデータD1,D2,D3,D4が格納されているとする。パリティディスクには、パリティデータP1が格納されているとする。パリティデータP1は(1)式により計算される。データディスク

Disk3が故障した場合には、Disk3に格納されていたデータD3は、故障していないディスクに格納されているデータから(2)式により計算され回復される。このような計算処理と再データ格納により故障ディスク中の全データが再生される。ただ、パリティデータは常に(1)式を満たさなければならないため、データディスク中のデータの更新時には(3)式に示される計算式にしたがってパリティデータをも更新する必要がある。例えばデータディスクDisk2中のデータD2を更新する場合、更新前の古いデータOld D2と古いパリティデータOld P1を読み出し、これらと新しい更新データNew D2との3者でXORをとったものを新しいパリティデータNew P1としてパリティディスクへ書き込む。このように、ディスクアレイでは、チェックディスクへの冗長なアクセスが必要となりディスクアレイでない場合に比べてそのスループットは低下する。

5.1.2 データの格納形式

次にデータのディスク装置への格納形式について述べる。文献[44]では、チェックデータの形式とユーザデータのディスク装置への格納形式によって5つの方式のディスクアレイを提案している。これら5つの方式を方式1~5とする(文献[44]ではRAID1~5としている)。以下では、これら5つの方式について説明を加える。図5.2は、方式2~5におけるチェックデータの形式とユーザデータのディスク装置への格納形式を表している。この図は文献[44]より抜粋したものである。ただし、以下に説明する方式4と5では、データディスクへ格納するデータは分割されずに、それぞれ独立な1個のディスク装置へ格納されるとする。したがって、Parity Striping[46]と呼ばれる方式に近い。

1) 方式1(RAID 1)は、1章ですでに述べたミラードディスクの場合である。同一内容をもつ2台のディスク装置を対にして用いる。1台はデータディスク、他の1台はチェックディスクとして用いられる。2台のディスク装置のどちらか一方が故障した場合でも他方のディスク装置からデータの読み書きが行える。また、故障していないディスク装置に格納されているデータを用いて、故障ディスクに格納されていたデータを回復できる。データの読み出しは2台のディスク装置のどちらかからでも行える。データの書き込みは、2台のディスク装置に対して常に行う必要がある。

2) 方式2(RAID 2)はハミングコードをチェックデータとして用いた場合である。図5.2に示されるようにデータディスクが4台の場合には、3台のチェックディスクが必要となる。転送ブロックはすべてのデータディスクに対して分割して格納される。図の例では、4つの転送ブロックa,b,c,dがあり、各転送ブロックは4つの小ブロックに分割され4台のデータディスクにそれぞれ格納される。このように転送データは分割されてディスク装置へ格納されるため、データの読み出し/書き込みはすべてのディスク装置にわたって行う必要がある。転送ブロックサイズが大きく分割された小ブロックがセクタサイズよりも大きな場合

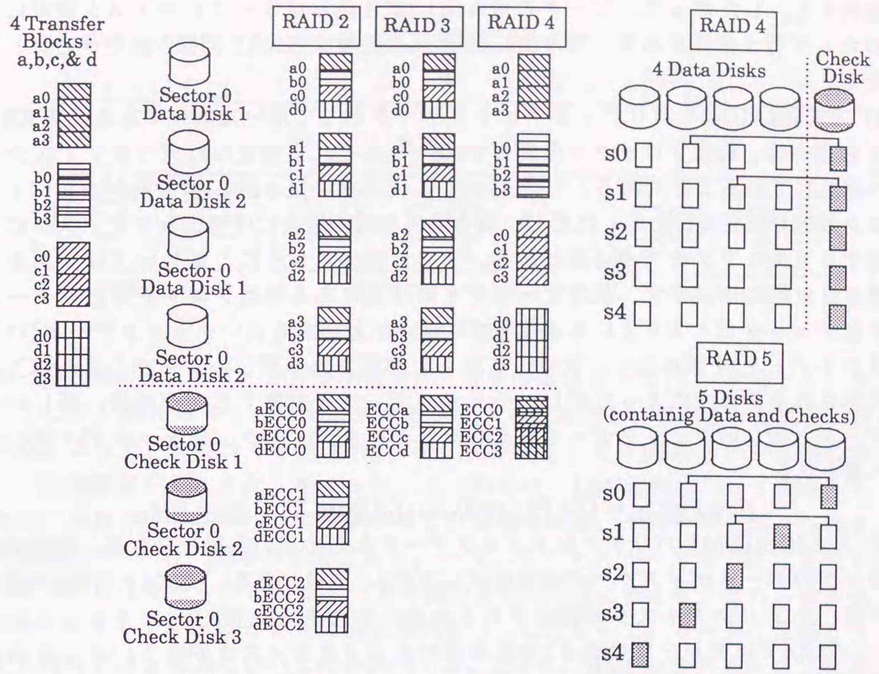


図5.2 ディスクアレイの各方式におけるディスク装置へのデータ格納形式(文献[44]より抜粋)

と、分割された小ブロックがセクタサイズよりも小さな場合では処理が異なる。前者の場合には、読み込み処理は各ディスクから該当する小ブロックを読み出すのみである。書き込み処理は各ディスクに該当する小ブロックを書き込み、それらのチェックデータをチェックディスクへ書き込むのみである。後者の場合、セクタ中には他の転送ブロックの分割された小ブロックも格納される。図5.2の例はこの場合を表している。セクタとはディスク装置アクセスにおけるデータの最小単位である。読み出し処理は各ディスクから該当する小ブロックを含む1セクタを読み出す。書き込み処理は各ディスクから該当する小ブロックを含む1セクタをまず読み出し、それらの中で更新されるべきブロックの該当する小ブロックのみ更新し、再びそれらを各データディスクへ書き戻す。この際、チェックデータもまず読み出し、更新されたデータに対応するようにそれらを更新し、再びそれらを各チェックディスクへ書き戻す。このように、チェックディスクへ対する冗長なアクセスが必要となる。

3) 方式3(RAID3)はパリティをチェックデータとして用いた場合である。数台のデータディスクからなる1組みのグループ(ディスクアレイ)に対して1台のチェックディスクが必要となる。図5.2の例では、4台のデータディスクに対して1台のチェックディスク(パリティディスクという)が設けられている。図に示されるように、転送ブロックを方式2と同様に分割して複数のデータディスクへ

格納する。したがって、データの読み出し/書き込みはすべてのディスク装置にわたって行う必要がある。読み出し/書き込み処理は方式2と同様の処理が必要となる。

4) 方式4(RAID4)もパリティをチェックデータとして用いた場合である。方式3との違いは、転送ブロックを分割せずに、それぞれを独立したデータディスクへ格納していることである。したがって、データの読み出し/書き込みは各ディスク装置で独立に行える。ただし、書き込み処理の場合にはチェックディスクに対する冗長なアクセスが必要となる。すでに述べたように、新しいブロックを書き込む際には、まず、現在データディスク上にある該当ブロックの古いデータとチェックディスク上にある該当ブロックに対応する古いチェックデータ(パリティデータ)を読み出す。次に、これらと更新される新しいデータとから以下に示される計算式によって新しいチェックデータを計算する。その後、新しいデータと新しいチェックデータをそれぞれデータディスク、チェックディスクへ書き込む。

$$\underline{\text{New Parity}} = \underline{\text{Old Parity}}\ \mathbf{XOR}\ \underline{\text{Old Data}}\ \mathbf{XOR}\ \underline{\text{New Data}}$$

5) 方式5(RAID5)もパリティをチェックデータとして用いた場合である。転送ブロックのデータディスクへの格納形式も方式4と同様である。方式4と方式5の違いは、チェックディスクが固定であるかないかである。図5.2に示されるように、方式4ではグループにつき1台の専用のチェックディスクがある。データの更新が頻繁に行われる場合には、チェックデータの更新も頻繁に生じる。したがって、チェックディスクへのアクセスが集中的に生じ性能が著しく低下する。方式5では、特定のチェックディスクはなく、グループ中のディスクにデータとチェックデータが分散されて格納される。したがって、チェックディスクへのアクセス集中は生じない。読み出し/書き込み処理は方式4と同様となる。

ディスクアレイの方式1~5のなかで、ディスク装置のみのスループットは方式1が最もよい。だが、方式1は全ディスク台数の半分が冗長ディスクであり、有効記憶容量(全ディスク台数におけるデータディスク台数の割合)は最低である。有効記憶容量が最良のものは方式3~5である。これらの中では、方式5が一番性能がよい。以下では、図5.3に示される一階層構成のDIMPを対象として、キャッシュ動作と処理性能を述べる。図5.3に示されるのは、8台のデータディスクに対して1台のパリティディスクを用いて1グループとしたディスクアレイ方式4に対応する構成である。8つのディスクアレイ·グループから成り、全データディスク台数は64台である。

5.2節　DIMPのディスクアレイ·システムにおけるキャッシュ動作

本節では、前節で解説したディスクアレイの各方式についてキャッシュ動作を示す。キャッシュデータの置き換え方式としてライトバック方式とライトスルー方式がある。ライトバック方式では、キャッシュヒット(リードヒット、ライトヒット)時にはキャッシュに対するデータの読み出し、書き込みが即座に行える。よって、キャッシュミス(リードミスとライトミス)時のデータ転送についてのみ説明する。また、キャッシュミス時には、LRU規則によって決められた捨てられるべきブロックが更新されている場合には、それをディスクへ書き戻す(ライトバック)処理が必要である。一方、ライトスルー方式では、リードヒット時にはキャッシュからデータの読み出しが即座に行える。また、書き込み動作はキャッシュのヒット/ミスに関係ない。したがって、リードミス時とライト時のデータ転送についてのみ説明する。図5.4はDIMPの各ディスクアレイ方式におけるディスク装置とディスクキャッシュ間のデータ転送を示している。

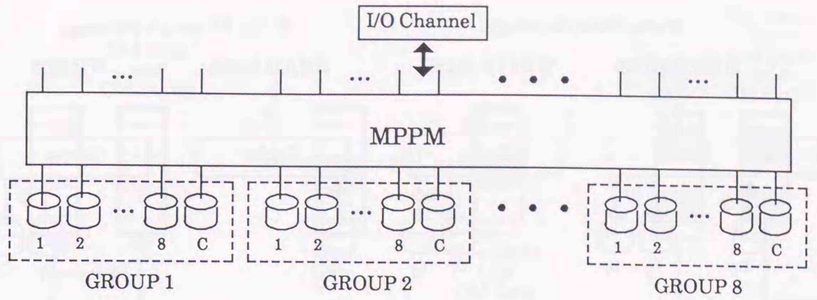


図5.3　DIMPの一階層構成

1) 方式1：ディスク装置への書き込みはデータディスクとそれと対になっているチェックディスクの2台のディスク装置に対して常に書き込み動作をする必要がある。図中のD_j,C_jは、それぞれj番目のデータディスク、j番目のチェックディスクを意味する。

2) 方式2：あるディスクのあるブロックを書き込みする時に、そのディスクが含まれるグループ(ディスクアレイ)中の他のディスクの該当するすべてのデータ(チェックデータを生成する組み)をアクセスする必要がある。そこで、フェッチブロックスをインターリーブし、1グループに渡って格納することにした。フェッチブロックスとは、キャッシュでリードミスが起きた時にディスク装置から読み込む連続した数ブロックをいう。例えば、フェッチブロック数を8ブロックとし、1グループ中のデータディスクの台数が8の時、各ディスク装置から1ブロックずつ読み込む。すべてのデータ転送はフェッチブロックス単位で行われる。図中のG_jは、j番目のグループ(ディスクアレイ)を意味する。

3) 方式3：方式2のキャッシュ動作と同様である。

4) 方式4：ディスクからの読み出しはデータディスク毎に行われる。書き込みは、データディスクの他にチェックディスクのデータも更新する必要がある。古いデータと古いチェックデータを読み出し、これらと更新された新しいデー

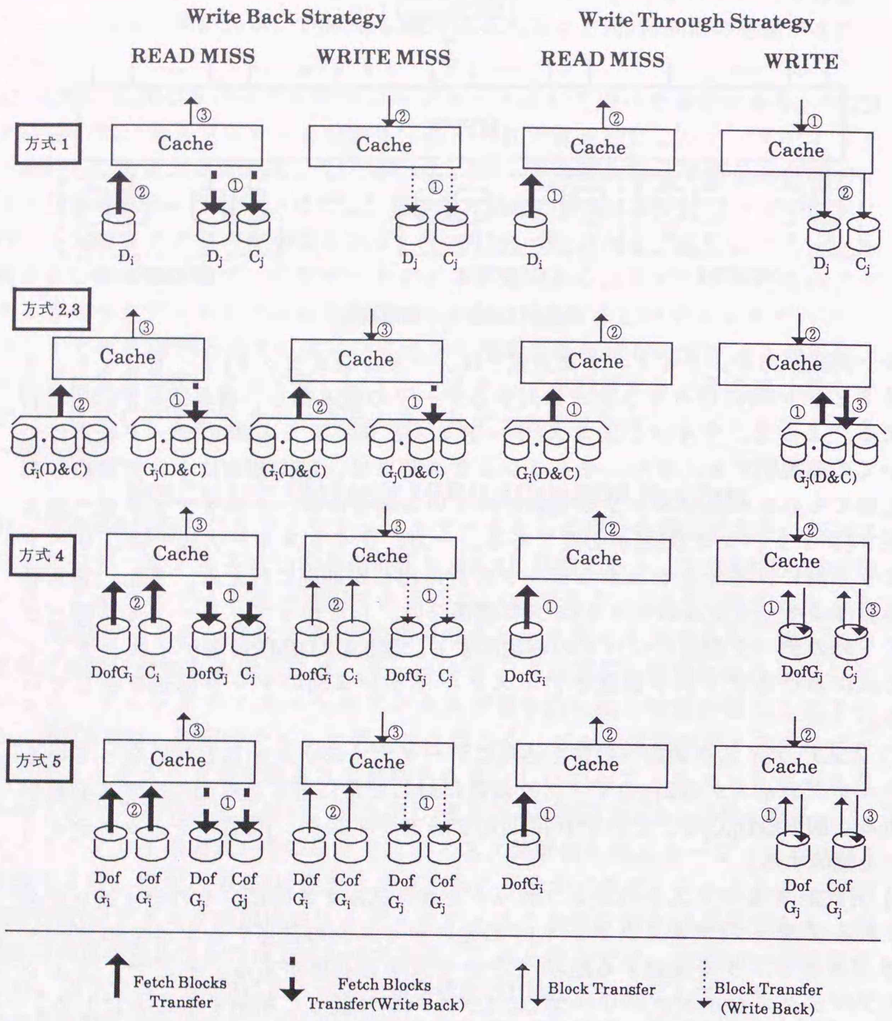


図5.4 DIMPの各方式におけるキャッシュ動作

タから新しいチェックデータを計算する。新しいデータと新しいチェックデータをディスクへ書き込む。ライトバック方式では、ライトバック時にディスクへの書き込み動作が必要となる。このとき、古いデータと古いチェックデータを読み込む必要がある。そこで、リードミスでフェッチブロックスがキャッシュに転送される時には、該当するチェックデータもあらかじめキャッシュへ転送しておく。リードミスあるいはライトミスで書き戻し処理が必要となっても、新しいデータと新しいチェックデータを書き込むだけでよい。図中のD of

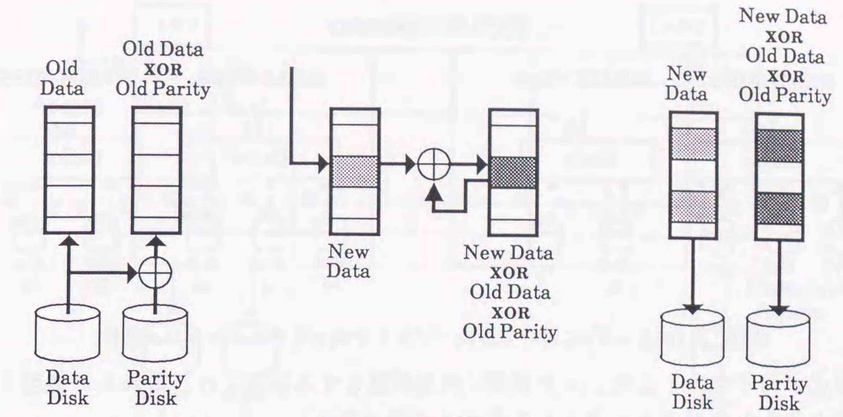


図5.5 ディスクアレイ方式4,5のライトバック動作時におけるディスク格納データの更新機構

G_jは、j番目のディスクアレイ·グループ中のデータディスクを意味する。C_jは、j番目のディスクアレイ·グループ中のチェックディスクを意味する。

5) 方式5：方式4のキャッシュ動作と同様である。方式5では、特定のチェックディスクはなく、1グループ中のディスクにデータとチェックデータが分散されている。したがって、上記のキャッシュ動作を行う場合には、分散の仕方はフェッチブロックス単位でなければならない。

上記ディスクアレイ方式4と方式5のライトバック動作時のディスクデータの更新機構を図5.5に示した。リードミスでディスク装置から数ブロックをキャッシュへ読み込む際、データディスク中の該当データを含む数ブロックとそれと対応するパリティディスク中の数ブロックを読み込む。図中(1)に示されるように、キャッシュ中にはその読み込んだ数ブロック·データの他、読み込みブロックとパリティブロックとの二者でXORをとった数ブロック(これを冗長データブロックと呼ぶ)が格納される。すでに、このデータブロックに対応する冗長データブロックがキャッシュに存在していた場合には、パリティブロックの読み込み処理は不要である。ライトヒットでキャッシュデータが更新される際には、この冗長データブロックも図中(2)のように更新される。更新されたブロックがキャッシュから追い出される際には、図中(3)のようにデータディスクとパリティディスクに対して書き戻し処理が行われる。ただし、キャッシュ中にこの冗長データブロックに対応する他のデータブロックが存在している場合には、パリティディスクへの書き込みは行われない。以後の5.3節で示す性能評価では、キャッシュミスで数ブロックをキャッシュへ読み込む際には、パリティディスク中の対応する数ブロックを必ず読み込むものとした。また、データをキャッシュからディスク装置へ書き戻す際にも、パリティディスクへの書き込

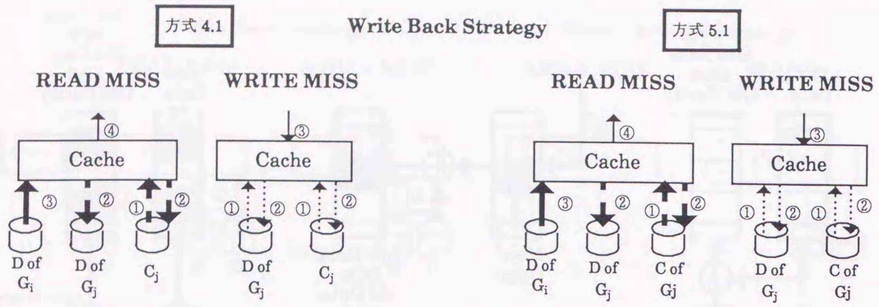


図5.6 方式4および5においてスループットを向上させるキャッシュ動作

みを必ず行うものとした。より厳密に性能評価をする場合には、これらの処理が必要かどうかをパラメータとして与える必要がある。

以上で説明した処理では、冗長データブロックをキャッシュに格納するために、実際にキャッシュに格納される(データディスク中の)データ量はキャッシュ容量より小さくなるというデメリットがある。また、データがキャッシュから捨てられる際に、それが更新されていない場合には書き戻し処理の必要がなく、チェックデータをキャッシュへ読み込んだ処理が無駄となる。よって、スループットがわずかに悪くなる。そこで、ライトバック時にだけ古いデータの読み出しと新しいデータの書き込みを行うようにしたのが図5.6に示されるキャッシュ動作である。この場合には、上記の場合に比べてわずかにスループットが良くなる。これらの方式を方式4.1、方式5.1とする。ただし、この場合にも更新されたブロックの古いデータをキャッシュ中で保持する必要があり、実際にキャッシュに格納されるデータ量はキャッシュ容量よりも小さくなる。

一方、図5.7は、従来のディスク·サブシステムのディスクアレイ方式2,3の場合のディスク装置とディスクキャッシュ間のデータ転送を示している。方式2,3以外はDIMPの場合と同様のデータ転送となる。従来のキャッシュ付きディスク·サブシステムでは、ディスク装置とディスクキャッシュ間のデータ転送がボト

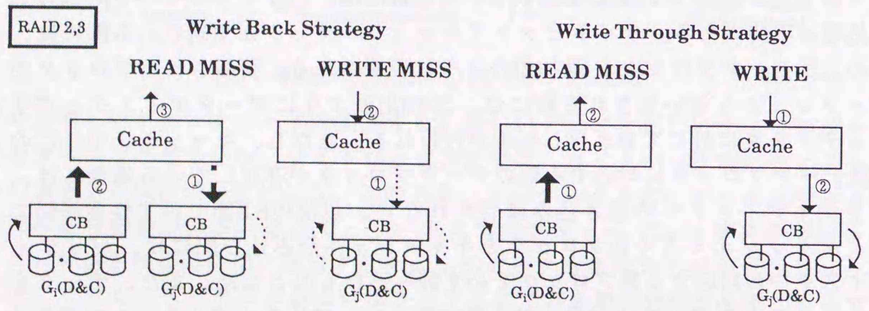


図5.7 従来のディスク·サブシステムにおけるRAID2,3のキャッシュ動作

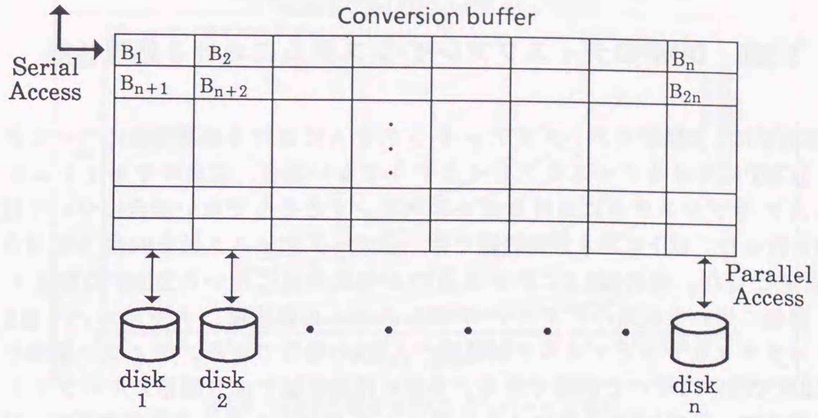


図5.8 コンバージョンバッファを介したデータ転送

ルネックとなっている。そこで、方式2,3では、図5.8に示されるようなコンバージョンバッファを介してディスクキャッシュとデータ転送を行うようにした。チェックディスクに対する冗長なアクセスをキャッシュから直接することなくコンバージョンバッファから行うことにより、ディスク装置とディスクキャッシュ間のデータ転送量を減らすことができる。これによりスループットが向上する。ディスクアレイ方式2,3の場合には、グループ内のすべてのディスク装置から並列にデータアクセスが行われる。また、3章図3.1に示されるように、従来のキャッシュ付きディスク·サブシステムでは、各ディスク装置が専用のパスでディスクキャッシュと接続される構成ではない。したがって、一般にはコンバージョンバッファ[51]を用いることでディスクアレイの1グループを1台のディスク装置のように扱うものと考えられる。

コンバージョンバッファは、ディスクキャッシュからのディスク装置に対する連続したデータの読み出しと書き込みを、複数のディスク装置に対する並列なデータの読み出しと書き込みへ変換する。各ディスク装置は同期して動作するものとする。チェックデータの更新は、このコンバージョンバッファとそれと接続されているディスク装置間で行われるものとする。したがって、ディスクキャッシュから見た場合には、コンバージョンバッファとそれと接続される複数のディスク装置は1台のディスク装置として扱われる。

5.3節　DIMPのディスクアレイ･システムにおける処理性能

本節では、DIMPのディスクアレイ･システムにおける処理性能について述べる。DIMPにおけるディスクアレイとそうでない場合、従来のキャッシュ付きディスク･サブシステムにおけるディスクアレイとそうでない場合について性能評価を行った。以下に示す性能評価では、全データディスク数を64台(全記憶容量を)固定とした。先の図5.3に示されるのが性能評価に用いたDIMPの構成である。評価に用いた従来のディスク･サブシステムの構成は、チャネルパス数8、ディスクキャッシュとディスク装置間パス数4の場合である。ディスク装置の性能は3章で用いたものと同様である。本章の性能評価では、限界スループットのみを求めた。先の4章で述べたようにディスク･サブシステムを各処理要素へ分解し、各処理要素での利用率が100%になる時のスループットを限界スループットとした。具体的な計算式は付録Ⅱに示されている。

5.3.1　評価結果

以下の図5.9、図5.10、図5.11、図5.12に示されたグラフはキャッシュ動作をそれぞれライトバックおよびライトスルー方式とした場合において、キャッシュのヒット率に対する1ディスク装置あたりのスループットを表している。1ディスク装置あたりのスループットとは、ディスク･サブシステム全体の限界スループットをそのシステムを構成しているディスク装置の台数で割ったものである。図5.9および図5.10はDIMPのライトバックおよびライトスルー方式の性能である。グラフの添字D1~4,41,5,51はディスクアレイ方式1~4,4.1,5,5.1の場合、D0はディスクアレイでない場合をそれぞれ表している。図5.11および図5.12は従来のディスク･サブシステムのライトバックおよびライトスルー方式の性能である。グラフの添字C1~4,41,5,51はディスクアレイ方式1~4,4.1,5,5.1の場合、C0はディスクアレイでない場合をそれぞれ表している。

表5.2は、DIMPのディスクアレイでない場合とディスクアレイ方式5.1の場合、従来のディスク･サブシステムのディスクアレイでない場合とディスクアレイ方式3の場合において、キャッシュのヒット率が80%のときの1ディスク装置あたりのスループットを挙げたものである。同様に、表5.3はキャッシュのヒット率が20%のときの1ディスク装置あたりのスループットを挙げたものである。単位はIO requests/sec/diskである。図5.9から図5.12の各グラフより、よい性能が得られる一部の方式について、その性能を列挙したものである。

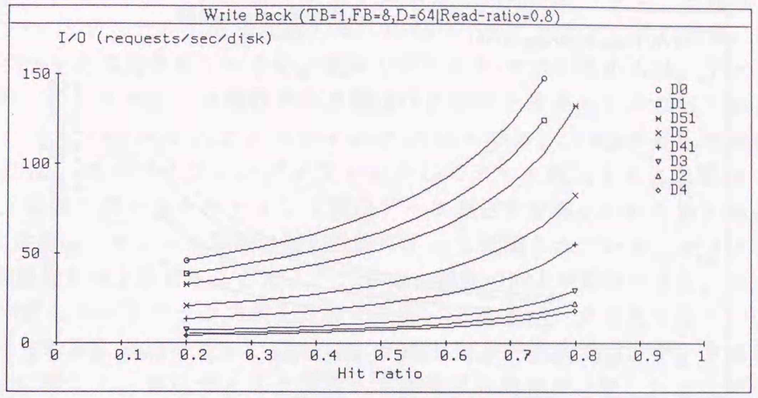


図5.9　DIMPの処理性能(ライトバック方式)

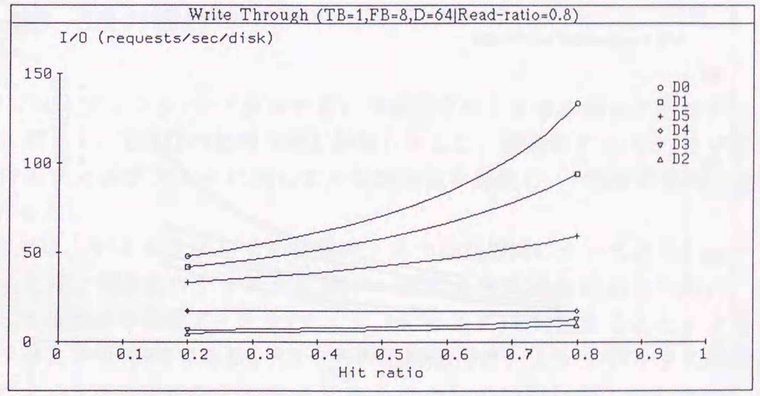


図5.10　DIMPの処理性能(ライトスルー方式)

5.3.2　考察

ディスクアレイの場合とそうではない場合のいずれにおいても、DIMPの性能は従来のディスク･サブシステムよりも優れている。特に、ヒット率が低い時のDIMPの性能は、ヒット率が高い時の従来のディスク･サブシステムの性能よりもよい。DIMPの性能は、従来のディスク･サブシステムの性能の6~7倍程度の性能が得られる。ディスクアレイの場合とそうではない場合では、ディスクアレイの方が性能が劣っている。これは、チェックディスクに対する冗長なアクセスが必要となるためである。ただ、DIMPのディスクアレイ方式5あるいは方式5.1では、ディスクアレイでない従来のディスク･サブシステムよりも良い性能を示している。すなわち、DIMPは、ディスクアレイ方式5,5.1により信頼性が上が

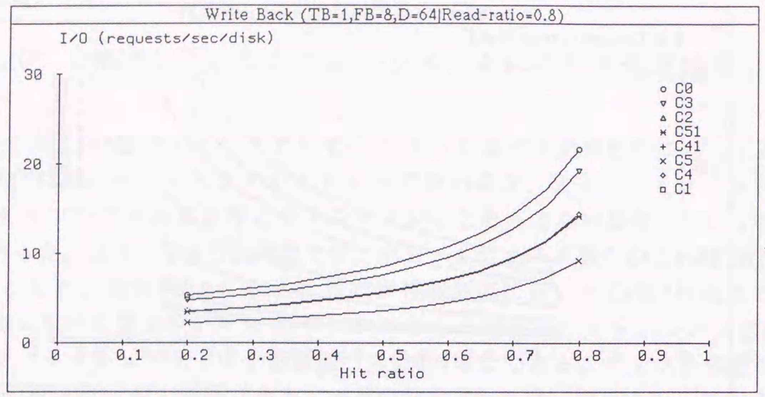


図5.11　従来のディスク･サブシステムの処理性能(ライトバック方式)

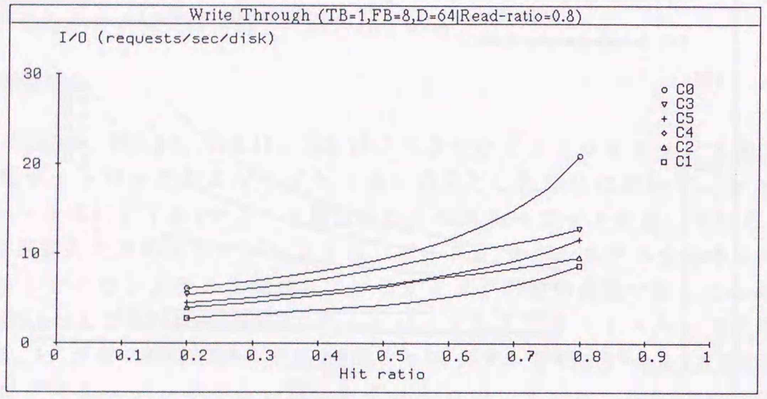


図5.12　従来のディスク･サブシステムの処理性能(ライトスルー方式)

表5.2　性能比較1(キャッシュヒット率80%)

	D0	D51	C0	C3
ライトバック	186	133	22	19
ライトスルー	134	60	21	13

表5.3　性能比較2(キャッシュヒット率20%)

	D0	D51	C0	C3
ライトバック	46	33	5.4	4.8
ライトスルー	48	33	6.2	5.5

るとともに、従来のディスク･サブシステムよりも高いスループットを達成するディスク･サブシステムであると言える。ただし、チェックデータの計算に要するオーバヘッドは含まれていない。従来のディスク･サブシステムは、ディスク装置とディスクキャッシュ間のデータ転送バスがボトルネックとなっている。従来のディスク･サブシステムのディスクアレイでは、方式3が最も性能がよい。これは、コンバージョンバッファを介してデータ転送することにより、ディスク装置とディスクキャッシュ間のデータ転送量が減少したためである。DIMPの性能は、ディスク装置の処理性能によって制限されている。ディスク装置の処理性能を向上させることにより、さらに性能の向上が期待できる。ライトスルー方式とライトバック方式を比較すると、ライトバック方式のほうがよい性能が得られることが分かる。これは、ライトスルー方式では、データの書き込み要求に際して、常にディスク装置へも書き込み処理が必要となるためである。

5.4節　5章の要約

本章では、ディスク･サブシステムの信頼性を向上させる構成であるディスクアレイに関して、DIMPの処理性能を評価し示した。従来のディスク･サブシステムにおけるディスクアレイに関しても処理性能を評価し、DIMPの処理性能との比較を行った。

5.1節では、ディスクアレイの故障ディスク回復機構について述べた。ディスクアレイとは、複数台のディスク装置(データディスク)を組みにして用い、それらに対して少数台の冗長ディスク(チェックディスク)を付加することにより信頼性を上げるというものである。データの格納形式とチェックデータの格納形式の違いによる5つの方式のディスクアレイについて説明した。これらは、文献[44]に挙げられているものである。

5.2節では、5.1節で述べた5つの方式のディスクアレイにおけるDIMPのキャッシュ動作を述べた。また、従来のディスク･サブシステムのディスクアレイにおけるキャッシュ動作に関しても説明を加えた。

5.3節では、5.2節で説明したキャッシュ動作を基に、DIMPのディスクアレイにおける性能と従来のディスク･サブシステムのディスクアレイにおける性能の評価を行った。ディスク装置あたりのスループットについて、従来のディスク･サブシステムにおけるディスクアレイの性能とDIMPの性能とを比較した。DIMPは従来のディスク･サブシステムの6~7倍程度のスループットが得られることが示された。

前3章で述べたように、従来のキャッシュ付きディスク･サブシステムでは、ディスク装置とディスクキャッシュ間のデータ転送幅がボトルネックとなっている。ディスクアレイにより信頼性を上げようとする場合にはチェックディス

クに対する冗長なアクセスを必要とし、ディスク装置とディスクキャッシュ間のデータ転送パスがさらにボトルネックとなる。DIMPでは、ディスク装置とディスクキャッシュ間のデータ転送パスが多数ありボトルネックとならない。DIMPは、ディスクアレイにより信頼性が向上すると共に、高スループットが得られるディスク･サブシステムであると言える。

第6章 結論

本章では、本論文の結論を述べる。まず、本研究の目的と成果の概要を顧みる。その後、2章以下で述べた内容を概観し、本研究の成果をまとめる。また、今後の課題について述べる。

近年、磁気ディスク装置の記憶容量は大容量化の傾向にある。だが、その処理速度の向上はほとんどみられない。一方、ディスク･サブシステムと接続される上位装置(主記憶装置、中央処理装置)の処理性能は飛躍的に向上している。上位装置の処理速度の高速化に伴い、ディスク･サブシステムは頻繁にアクセスされるようになる。したがって、ディスク･サブシステムの高スループット化が望まれる。高スループットを達成するディスク･サブシステムのアーキテクチャを示し、その処理性能を明らかにすることを本研究の目的とした。

ディスク･サブシステムを高スループット化するためには、ディスクキャッシュのデータ転送幅を上げる必要がある。主記憶装置の高速な処理速度と機械動作を含む磁気ディスク装置の低速な処理速度の差を緩和し、処理性能を向上させるための緩衝メモリがディスクキャッシュである。すでに、マルチポート･ページメモリ(MPPM)と呼ばれるメモリをディスクキャッシュとして用いることが提案[1]されている。MPPMは、複数の入出力ポートをもち、どのポートからも並行にブロック単位でデータの入出力が行えるメモリである。だが、その処理性能についての報告はない。また、MPPMをバッファメモリとして用いた知識ベースマシンの研究[67]もある。MPPMを用いることにより高スループットが得られることが示されている。だが、MPPMをディスクキャッシュとして階層的に用いるという提案は今までない。そのキャッシュ動作と処理性能についての報告もない。

本論文では、MPPMをディスクキャッシュとして階層的に用いたディスク･サブシステムDIMPのアーキテクチャを提案した。DIMPの種々の構成とキャッシュ動作における処理性能について述べた。シミュレーションによる性能評価から、DIMPでは、ディスク装置の処理性能あるいは入出力チャネルのデータ転送幅により制限されるまでの高いスループットが得られることを示した。階層化されたディスクキャッシュ構成において、異なる入出力チャネルに独立に接続される複数のキャッシュメモリがある場合、キャッシュデータのコヒレンシ問題が生じる。DIMPの二層全接続型構成と呼ばれる構成では、キャッシュデータのコヒレンシ問題がある。DIMPの二層全接続型構成に関して、キャッシュデータのコヒレンシを考慮した4種のキャッシュ動作における処理性能を示し、それらの特性を明らかにした。さらに、信頼性を向上させるディスク構成である

ディスクアレイに関して、MPPMをディスクキャッシュとして用いた場合の処理性能と特性を明らかにした。

本研究の遂行において性能評価は不可欠である。種々の構成の対象システムについてそれらのモデリングが容易に行え、シミュレーション中の対象モデルの状態が容易に把握でき、対話的にシミュレーションが行えるシミュレータが必要である。このようなシミュレータを対話型グラフィカルシミュレータと呼んだ。このような機能をもつシミュレーションシステムの開発を本研究の初期目的し、対話型グラフィカルシミュレータを開発するツール群を提供するシステムとしてFESを開発した。さらに、FESを用いて待ち行列網を基本とした対話型グラフィカルシミュレータを開発した。VLSI設計用のCADをはじめとして対話型グラフィカルシミュレータは多数開発されている。だが、これらではその対象分野が固定されており、任意の動作機構をもつ新たな構成要素の追加が行えない。FESは、MV(Model-View)モデリングと呼ぶモデリング手法を採用したことで、任意のオブジェクトを新たな構成要素として追加できる。MVモデリングとは、画面上に表示される形状を定義した視覚表示部(View)と内部動作機構を定義した動作記述部(Model)に分けてオブジェクトの定義を行うモデリング手法である。FESでは種々の機能をもつ視覚表示部をあらかじめ用意している。動作記述部のみを定義し登録することで、任意の動作機構をもつ対象システムの構成要素を容易に追加できる。

本論文では、MVモデリングをはじめとして、FESが提供する種々の機能と特徴を述べ、対話型グラフィカルシミュレータの開発においてそれらがどのように用いられているかを述べた。

以下では、各章で述べられた内容と得られた結論を概観すると共に、今後の課題について述べる。

2章では、FESシミュレーションシステムの仕様および機能について述べた。MVモデリングについて説明し、FESの特徴であるビジュアル合成機能とビジュアルインスペクタ機能の実現に、それがどのように用いられているかを示した。また、FESを用いて開発されたシミュレータである待ち行列網シミュレータCABを挙げ、FESの有用性を示した。特に、ビジュアル合成による対象モデルの構成の視覚的定義機能と、ビジュアルインスペクタによる対象モデルの状態･動作のグラフィカルな表示機能が、CABにどう活かされているかを示した。

FESは以下の機能を提供することを示した。

(1) シミュレーションの対象モデルの構成を視覚的に定義する機能
(2) 対象モデルの状態や動作状況を視覚表示する機能
(3) シミュレーション結果を種々の形式で表示する機能
(4) 対話型操作のための統合環境

FESの開発目的は、計算機アーキテクチャの研究を行う上での研究支援ツールの開発であった。MVシステムにより、シミュレーション対象の構成要素の動作

機構のみをプログラミングし定義･登録することで、種々の動作機構をもつ基本モデルを利用可能となる。FESを用いて開発された待ち行列網シミュレータCABは、種々の動作機構をもつ基本モデルを用意している。ディスク･サブシステムなどの複雑な動作機構をもつ対象システムのシミュレーションも、全くプログラミングせずに行うことができた。FESは、計算機アーキテクトにとって有益な研究支援ツールであるといえる。

3章では、MPPMをディスクキャッシュとして階層的に用いたディスク･サブシステムDIMPのアーキテクチャについて述べた。DIMPの構成と基本的なキャッシュ動作について解説した。また、従来のキャッシュ付きディスク･サブシステムの構成と動作についても解説し、そのボトルネックを指摘した。待ち行列網シミュレータCABにより性能評価を行った結果から、従来のキャッシュ付きディスク･サブシステムの性能とDIMPの性能を比較した。従来のキャッシュ付きディスク･サブシステムでは、ディスク装置とディスク･キャッシュ間のデータ転送パス数がボトルネックであり、スループットはキャッシュのヒット率に依存することを示した。DIMPでは、MPPMをディスクキャッシュとして用いることにより、この間のパス数を十分大きくでき、ボトルネックとならない。DIMPでは、以下の特徴があることを述べた。

(1) キャッシュのヒット率が低い場合にも、チャネルの転送幅あるいは、ディスク装置の処理性能によって制限されるまでの高いスループットが得られる。
(2) RPSミスの発生を抑えることができ、待ち時間を小さくできる。
(3) 構成上の自由度が高く、システム拡張が容易である。

実際にDIMPを実現する場合には、構成･動作に関してより詳細な検討が必要である。また、コストについての検討も必要である。

将来、ディスク･サブシステムの記憶容量は、ますます大きくなり、しかも高いスループットが望まれるようになる。データの転送速度を上げることで、スループットを上げることは可能である。けれども、処理速度の速いメモリデバイスを用い、短時間で複雑な制御をしなければならない。しかも、キャッシュのヒット率が低い場合には、ディスク･ドライブに頻繁にアクセスしなければならず、機械的に動作するディスク装置の処理速度を上げることは難しい。したがって、データの転送速度を上げることで、スループットを向上させるには限界がある。このように考えると、MPPMをキャッシュメモリとして用いたDIMPは、スケーラブルな性能が得られ、将来のディスク･サブシステムとして有望であると思われる。

また、データベース･システムなどで、データの検索を高速に行うためには、ポインタでつながれた関連あるデータ群を一度に読み込める必要がある。これらデータ群は、通常複数のディスク装置にわたって格納されている。したがって、複数のディスク装置から並列にデータのアクセスが行える場合、高速なデータ検索が可能である。このような点から、複数のディスク装置から並列に

データのアクセスが行えるDIMPは、データベース･マシンのディスク･サブシステムとしても有望であると思われる。

4章では、MPPMをキャッシュメモリとして階層的に用いたディスク･サブシステムDIMPについて、特に、上下層に同数のMPPMを配した二層全接続型構成に関して述べた。キャッシュデータのコヒレンシを考慮した4種のキャッシュ動作とそれらの処理性能を示した。M/M/1待ち行列モデルを用いて解析的に得られた評価性能からそれらの特徴を明らかにした。

キャッシュデータの置き換え方式として、上下層のMPPMそれぞれについて、ライトスルー(WT)方式とライトバック(WB)方式が採用でき、4種のキャッシュ動作(WB/WB方式、WT/WB方式、WB/WT方式、WT/WT方式)が考えられる。解析的に得たこれらの処理性能より、限界スループットおよび平均応答時間を比較し以下の結果を得た。

･限界スループット

WB/WB＞WT/WB＞WB/WT＞WT/WT　(良い＞悪い)

･(低トランザクション 域での)平均応答時間

WB/WT＞WT/WT＞WT/WB＞WB/WB　(良い＞悪い)

これらの結果と実現の容易さから、WT/WB方式が最適な方式であると考えられる。この評価では、アクセス集中に関しては考慮していない。より現実的な場合の性能を見るためにはアクセス集中を考慮して評価する必要がある。

5章では、ディスク･サブシステムの信頼性を向上させる構成であるディスクアレイに関して、DIMPの処理性能を示した。ディスクアレイとは、故障ディスクに格納されていたデータを、他の故障していない複数のディスクに格納されたデータから回復することにより信頼性を向上させるものである。ユーザデータが格納される複数のディスク装置(データディスク)に対して、少数台の冗長ディスク(チェックディスク)が必要となる。ディスク装置あたりのスループットについて、従来のディスク･サブシステムにおけるディスクアレイの性能とDIMPの性能とを比較した。DIMPは従来のディスク･サブシステムの6~7倍程度のスループットが得られることが示された。

従来のキャッシュ付きディスク･サブシステムでは、ディスク装置とディスクキャッシュ間のデータ転送幅がボトルネックとなっている。ディスクアレイにより信頼性を上げようとする場合にはチェックディスクに対する冗長なアクセスを必要とし、ディスク装置とディスクキャッシュ間のデータ転送幅がさらにボトルネックとなる。DIMPでは、ディスク装置とディスクキャッシュ間のデータ転送パスが多数ありボトルネックとならない。このような点から、MPPMをディスクキャッシュとして用いることによりディスク装置とディスクキャッシュ間のデータ転送幅を上げるという試みは意義があると思われる。ま

た、MPPMをディスクキャッシュとして用いた場合のディスクアレイの性能については、今まで報告がない。その点で本性能評価は意義があると思われる。

ディスクアレイについては、チェックデータ(チェックディスクに格納されるデータ)の格納形式が種々考えられている。5章では、文献[44]に挙げられているディスクアレイの5つの方式についてのみ性能評価を行った。他の方式のディスクアレイに関しても性能を明らかにする必要がある。さらに、MPPMをディスクキャッシュとして用いた場合に最適なディスクアレイの方式を検討する必要がある。

最後に、本論文では、ディスク･サブシステムの高スループット化を研究目的とし、ディスクキャッシュとしてMPPMを階層的に用いることを提案した。種々の構成についてキャッシュ動作を検討し、それらの評価性能を示し特性を明らかにした。本研究の遂行上不可欠である性能評価のために、FESシミュレーションシステムの開発を本研究の初期目的とした。さらに、FESを用いて待ち行列網シミュレータCABを開発した。ディスク･サブシステムの性能評価ツールとして実際に使用し、その有用性を示した。

謝辞

本論文を書くにあたり、御指導下さいました田中譲教授に深く感謝致します。田中先生には、常に問題意識を持ち、問題解決のために努力することの大切さ等、多くの事を教えて頂きました。また、深く、豊富な知識をもとにした、鋭い御指摘、御助言を多数頂きましたことを心より感謝致します。

本研究の遂行において、多くのご助言を頂いた日立製作所小田原工場、宮崎道生氏に深謝いたします。宮崎氏には、ディスク装置の構成･動作など、本研究を遂行する上で必要となる種々の知識と情報をお教え頂きました。ここに感謝の意を表します。

さらに、山本章博講師に謝意を表します。山本先生には、研究に望む姿勢等、研究を進める過程で多くの御助言を頂きました。ここに感謝を述べます。また、応用制御工学講座のみなさまにも、多数の御助言を頂きましたことを深く感謝致します。

文献

[1] Y. Tanaka, “A Multiport Page-Memory Architecture and A Multiport Disk-Cache System”, New Generation Computing, 2, pp. 241-260, OHMSHA, Tokyo(February 1984).

[2] Special issue, IEEE COMPUTER, Vol. 7,No. 12, pp. 28-67(1974).

[3] S. S. Ching and J. H. Tracey, ”An Interactive Computer Graphics Language for the Design and Simulation of Digital Systems”, IEEE COMPUTER, Vol. 10, No. 6, pp. 35-41(1977).

[4] M.G.Walker and J.McGregor, “Computer-Aided Engineering for Analog Circuit Design”,I EEE COMPUTER, Vol. 19, No. 4, pp. 100-108(1986).

[5] E. F. Girczyc and TaiLy, “STEM: An IC Design Environment Based on the Smalltalk Model-View-Controller Construct”, 24th IEEE Design Automation Conference, pp.757-763(1987).

[6] 梅村恭司著,竹内郁雄監修, “Smalltalk-80入門”,ソウトウェアライブラリ=4, サイエンス社(1986).

[7] H. Brown, C. Tong, and G. Foyster, “Palladio:An Exploratory Environment for Circuit Design” ,IEEE COMPUTER, Vol. 16, Vo. 12, pp. 41-56(1983).

[8] Pieter S. van der Meulen, “INSIST:Interactive Simulation in SmallTalk”, OOPSLA ’87 Proceedings, pp.366-376,October(1987).

[9] 杉本明,阿部茂, “オブジェクト指向言語VEGAMESによる構造レベル･ハードウェアのモデル化”, コンピュータソフトウェア, Vol. 3, No. 3, pp. 71-85(1986).

[10] 杉本明,阿部茂ほか, “ハードウェア動作記述言語:ALHARD”, 情報処理学会第37回(昭和63年後期)全国大会講演論文集(III), pp.1743-1748(1988).

[11] 小島泰三,杉本明,阿部茂, “ハードウェアシミュレーションシステムALHARDービジュアルインタフェースの拡張”, 情報処理学会第40回(平成2年前期)全国大会講演論文集(III), pp.1359-1360(1990).

[12] 藤田友之,久富雄二ほか, “電気系EWSシステムPROCEED”,情報処理学会第37回(昭和63年後期)全国大会講演論文集(III), pp.1784-1795(1988).

[13] 小池田恒行ほか, “論理回路から簡単な計算機レベルまで適用可能な教育用シミュレータ”, 情報処理学会第35回(昭和62年後期)全国大会講演論文集(III), pp.2673-2675(1987).

[14] Rajiv Gupta, Wesley H. Cheng, Rajesh Gupta, Ido Hardonag, and Melvin A. Breuer, "An Object-Oriented VLSI CAD Framework", IEEE COMPUTER, Vol. 22, No. 5, pp. 28-37(1989).

[15] 神戸尚志,谷貞宏ほか,"VLSIレイアウト設計のための統合化支援システム", 情報処理学会論文誌, Vol. 31, No.3, pp. 351-360(1990).

[16] B.Melamed and R.J.T.Morris, "Visual Simulation : The Performance Analysis Workstation",IEEE COMPUTER, Vol. 18, No. 8, pp.87-94(1985).

[17] Kathleen M.Nichols and John T.Edmark, "Modeling Multicomputer Systems with PARET", IEEE COMPUTER, Vol. 21, No. 5, pp. 39-48(1988).

[18] Giorgio Bruno and Alessandro Balsamo, "PETRI NET-BASED OBJECT-ORIENTED MODELLING OF DISTRIBUTED SYSTEMS", Proc. OOPSLA '86 ACM, pp.284-293, September(1986).

[19] Mack Alford, "SREM at the Age of Eight ; The Distributed Computing Design System", IEEE COMPUTER, Vol. 18, No. 4, pp. 36-54(1985).

[20] 山口英,下条真司,宮原秀夫, "分散処理システム評価シミュレータSEDS", 情報処理学会論文誌, Vol. 30, No. 6, pp. 752-760(1989).

[21] Richard h. Lathrop and Robert S. Kirk, "An Extensible Object-Oriented Mixed-Mode Functional Simulation System" ,22nd IEEE Design Automation Conference, pp. 630-636(1985).

[22] 金井直樹, "シミュレーション・システムとしてのスプレッド・シートの機能拡張", 情報処理学会論文誌, Vol. 30, No.10, pp. 1335-1345(1989).

[23] F. Lakin, "Spatial Parsing for Visual Languages" in Visual Languages, edited by S.K. Chang, T. Ichikawa and P.A. Ligomenides, Plenum Publishing Corp.(1986).

[24] M. Edel, "The Tinkertoy Graphical Programming Environment", Proceedings of IEEE 1986, pp. 466-471(1986).

[25] T. Kurita and K. Tamura, "Dialog.I: an Iconic Programming System Based on Logic Programming", Bulletin of the Electrotechnical Laboratory, Japan, Vol. 48, No. 12, pp. 966-975(1984).

[26] A. Giacalone, M. C. Rinard, and T. W. Doeppner, "IDEOSY:An Ideographics and Interactive Program Description System", ACM SIGPLAN Notices, Vol. 19, No. 5, pp. 15-20(1984).

[27] S. Matwin and T. Pietrzykowski, "PROGRAPH: A Preliminary Report", Computer Language, Vol. 10, No. 2, pp. 19-126(1985).

[28] E. P. Glinert and S. L. Tanimoto, "Pict : An Interactive Graphical Programming Environment", IEEE COMPUTER, Vol. 17, No.11,pp.7-25(1984).

[29] Jeff Kramer, Jeff Magee, and Keng Ng, "Graphical Configuration Programming", IEEE COMPUTER, Vol. 22, No. 10, pp. 53-65(1989).

[30] 増田英孝,笠原宏, "Smalltalk-80における拡張MVCモデルとその応用",情報処理学会論文誌, Vol. 31, No. 2, pp. 259-267(1990).

[31] 布川博士,富樫敦,野口正一, "図形をシンタックスに持つ関数型言語", 情報処理学会第35回(昭和62年後期)全国大会講演論文集(I), pp. 683-685(1987).

[32] T. Lehr, Z. Segall, D. F. Vrsalovic, E. Caplan, A. L. Chung, and C. E. Fineman, "Visualizing Performance Debugging", IEEE COMPUTER, Vol. 22, No. 10, pp. 38-51(1989).

[33] 市川至,小野越夫,毛利友治, "プログラム可視化システム", 情報処理学会論文誌, Vol. 31, No. 12, pp. 1801-1811(1990).

[34] Y. Tanaka, "A Toolkit System for the Synthesis and the Management of Active Media Objects", Proc. 1st Int. Conf. on Deductive and Object Oriented Databases, Kyoto, Dec.(1989).

[35] L.D.Stevens, "The Evolution of Magnetic Storage", IBM Journal of Research and Development, Vol. 25, No. 5, pp. 663-675,Sept.(1981).

[36] J. M. Harker et al., "A Quarter Century of Disk File Innovation", IBM Journal of Research and Development, Vol. 25, No. 5, pp.677-689,Sept.(1981).

[37] R.E.Matick, "Impact of memory systems on computer architecture and system organization", IBM SYSTEMS JOURNAL, Vol. 25, pp. 274-305 (1986).

[38] T.Teorey and T.B. Pinkerton, "A comparative analysis of disk scheduling policies", Commun. ACM, Vol. 15, No. 3, pp. 177-184, Mar.(1972).

[39] N.C.Wilhelm, "An anomaly in disk scheduling: A comparison of FCFS and SSTF seek scheduling using an empirical model for disk accesses", Commun. ACM, Vol. 19, No. 1, pp. 13-17, Jan.(1976).

[40] R.Geist and S. Daniel, "A Continuum of Disk Scheduling Algorithms", ACM Transactions on Computer Systems, Vol. 5, No. 1, pp. 77-92(1987).

[41] 掛下哲郎,上林弥彦,"2次記憶アクセスに関する最適な並行処理スケジューラ", 電子情報通信学会論文誌D-I, Vol. J72-D-I, No. 3, pp. 196-204(1989).

[42] D. Bitton and J. Gray, "Disk Shadowing", VLDB 1988 proceedings, Morgan Kauffman, pp.331-338, Sept.(1988).

[43] W. N. Spencer, "Improving Disk Performance Via Latency Reduction", IEEE Transactions. on Computer, Vol. C-40, No. 1, pp. 22-30(1991).

[44] D. A. Patterson, G. A. Gibson, and R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)", Proceedings of the ACM SIGMOD Conference, pp.109-116 (1988).

[45] T. M. Olson, "Disk Array Performance in a Random IO Environment", ACM SIGARC, Vol. 17, No. 5, pp.71-77, Sept.(1989).

[46] Jim Gray, Bob Horse and Mark Walker, "Parity Striping of Disc Arrays : Low-Cost Reliable Storage with Acceptable Throughput", Proc. of the 16th VLDB Conference, pp. 148-161(1990).

[47] Jai Menon, "Comparison of Sparing Alternatives for Disk Arrays", Proc. of 19th Ann. Int. Symp. on Computer Architecture, pp. 318-329, May(1992).

[48] Mark Holland and Garth A. Gibson, "Parity Declustering for Continuous Operation in Redundant Disk Arrays", ACM ASPLOS V, pp.23-35(1992).

[49] G. A. Gibson, L. Hellerstein, R. M. Karp, R. H. Katz and D. A. Patterson, "Failure Correction Techniques for Large Disk Arrays", ACM SIGARC, Vol. 17, No. 5, pp. 123-132, Sept.(1989).

[50] Kenneth Salem,Hector Garcia-Molina, "DISK STRIPING", Int. Conf. on Data Engineering, IEEE, pp.336-342(1986).

[51] M. Y. Kim, "Synchronized Disk Interleaving", IEEE Transactions on Computer, Vol. C-35, No. 11, pp. 978-988(1986).

[52] A.L.NARASIMHA REDDY and PRITHVIRAJ BANERJEE,"An Evaluation of Multiple-Disk I/O Systems", IEEE Trans. on Computer ,Vol. 38, No. 12, pp. 1680-1690(1989).

[53] A.L.NARASIMHA REDDY and PRITHVIRAJ BANERJEE, "A STUDY OF PARALLEL DISK ORGANIZATIONS", ACM SIGARC, Vol. 17, No. 5, pp. 40-47, Sept.(1989).

[54] Richard R. Muntz and John C.S. Lui, "Performance Analysis of Disk Arrays Under Failure", Proc. of the 16th VLDB Conference, pp.162-173(1990).

[55] Y. Bard, "A model of shared DASD and multipathing", Commun. ACM, Vol. 23, No. 10, pp. 564-572(1980).

[56] 猪苗代勉,徳永威久ほか, "統合ディスクキャッシュシステム", 日本電気技報 No. 133, pp. 45-50(1980).

[57] 平野正信, "アクセス・ギャップを埋めるディスク・キャッシュの機能を見る", 日経コンピュータ, 1982年3月22日号, pp.71-85(1982).

[58] A. J. Smith, "Disk cache-miss ratio analysis and design considerations", ACM Trans. on Comput. Sys., Vol. 3, Aug.(1985).

[59] 桜井紀彦, "ディスクキャッシュのコスト性能比評価方式に関する一考察", 情報処理学会第35回(昭和62年度後期)全国大会講演論文集(I), pp.175-176(1987).

[60] 小畑征二郎,松沢茂,宮崎正俊,神山典,表俊夫, "ディスク・キャッシュの効果に関する一考察", 情報処理学会論文誌, Vol. 26, No. 6, pp. 1009-1016(1985).

[61] 畑下豊仁,志賀稔ほか, "UNIXワークステーションにおけるディスク・アクセス特性とディスク・キャッシュの考察", 情報処理学会論文誌, Vol. 28, No. 6, pp. 606-616(1987).

[62] 西垣通,山本彰, "順次アクセス入力処理におけるディスク・キャッシュ装置の効果解析", 情報処理学会論文誌, Vol. 25, No. 2, pp.313-320(1984).

[63] 三石彰純,宮地泰造,溝口徹夫, "バッファ内蔵型ディスク装置の性能評価", 電子通信学会論文誌, Vol.J67-D, No. 11, pp. 1301-1308(1984).

[64] 宮地泰造,三石彰純,溝口徹夫, "階層型ディスク・キャッシュ・サブシステムの性能評価", 電子通信学会論文誌, Vol.J68-D, No. 9, pp. 1609-1616(1985).

[65] 山本彰,坪井俊明,北嶋,本間繁雄,野沢正史, "記憶階層システムにおけるプリロードセット学習制御方式とキャッシュ付きディスク・サブシステムへの適用", 情報処理学会第42回(平成3年前期)全国大会講演論文集IV, pp. 45-46(1991).

[66] 物井秀俊,森田幸伯ほか, "マルチポートページメモリを用いた知識ベースマシンの並列制御方式と処理性能", 情報処理学会論文誌, Vol. 29, No. 5, pp.513-520(1988).

[67] 湯浅太一、萩谷昌己著、"Common Lisp入門"、<岩波コンピュータサイエンス>、1987.

[68] Hank Bromley,"LISP LORE:A GUIDE TO PROGRAMING THE LISP MACHINE",Kluwer Academic Publishers,1986.

[69] Nan C.Shu,"Visual Programming",Van Nostrand Reinhold ,1988.

[70] M.H.MacDougall,“Simulating Computer Systems Techniques and Tools”,Computer Systems Series,The MIT Press,1987.

[71] E.Gelenbe,I.Mitrani共著、秋丸春夫、橋田温監訳、”計算機システムの解析と設計”、オーム社、1988.

[72] M.Ajimone Marsan,G.Balbo,and G.conte, "Performance Model of Multiproceessor Systems", Computer Systems Series,The MIT Press,1986.

[73] W.ライシッヒ著、長谷川健介、高橋宏治訳、”ペトリネット理論入門:並列同時進行の表現と解析“、シュプリンガー・フェアラーク東京株式会社、1988.

[74] 岡田義広、田中譲、“対話型ビジュアル・シミュレータ:グラフィカルなモデル表現とモデル合成によるシステム記述”、情報処理学会第37回(昭和63年後期)全国大会講演論文集(I),pp.194-195,1988.

[75] 岡田義広、田中譲、”対話型ビジュアル・シミュレータCAB”、情報処理学会、計算機アーキテクチャ研究会報告77-5、pp.35-42、1989.

[76] 日立製作所、HITAC(H-8538-C3ディスク制御装置、H-6585ディスク駆動装置、H-8598ディスク駆動装置)解説書、資料番号8080-2-094-10

[77] Brown,D.t., et al.,“Channel and Direct Access Device Architecture”,IBM Syst. j.,3,pp.186-199(1972).

[78] NIKKEI ELECTRONICS 1992.3.2.(no.548),pp. 142-148.

[79] Goodman J.R. : “USING CACHE MEMORY TO REDUCE PROCESSOR-MEMORY TRAFFIC”, ACM 10th Annual International Symposium on COMPUTER ARCHITECTURE, pp. 124-131(1983).

[80] Swazey P. and Smith A.J. : “A Class of Compatible Cache Consistency Protocols and their Support by the IEEE Futurebus”, IEEE 13th Annual International Symposium on COMPUTER ARCHITECTURE, pp. 414-423(1986-06).

[81] A.L.Narasimha Reddy, “A Study of I/O System Organizations”, Proc. of 19th Ann. Int. Symp. on Computer Architecture, pp.308-317,May, 1992.

論文目録

(1) 岡田義広,田中譲, :マルチポート・ページメモリをキャッシュとして用いた高性能ディスク・サブシステムのアーキテクチャと処理性能, 情報処理学会論文誌, Vol.33,No.12,pp.1625-1636 (1992)

(2) 岡田義広,田中譲, : 視覚的シミュレータの開発支援システム, 情報処理学会論文誌, Vol.32,No.6,pp.766-776(1991)

(3) 岡田義広,田中譲, : マルチポート・ページメモリを階層的に用いたディスクキャッシュのキャッシュ動作と処理性能, 電子情報通信学会論文誌D-I, Vol. J75-D-I, No.11, pp.1037-1047(1992)

付録 I

キャッシュデータのコヒレンシを考慮した4つのキャッシュ動作における性能評価式を以下に挙げる。

各処理要素(I/Oチャネルと上層のMPPM間の転送パス、上層と下層のMPPM間の転送パス、ディスク装置)での利用率から、先の図4.9で示されるように平均待ち時間が求まる。利用率を求めるための平均サービス時間と限界スループットを以下に示す。ディスク･サブシステム全体の限界スループットは、各処理要素における限界スループットの最小値となる。また、平均応答時間は、各データ転送(リードヒット、リードミス、ライトヒット、ライトミス)における処理時間とその時の場合の確率とを積算したものすべてを加えて求められる(Average Response Time = RH + RM + WH + WM)。

簡単のために、以下の変数を導入する。

●各処理要素での平均サービス時間と限界スループットを求める式中で用いられる変数。

I/Oチャネルでのブロックデータ転送における平均サービス時間 : STcp＝Tb＋Tpw.

上下層のMPPM間パスでのブロックデータ転送における平均サービス時間 : STip＝Tb＋Tpw.

ディスク装置のブロックデータ転送における平均サービス時間 : STd-b＝Tsw＋Tb＋Tpw.

ディスク装置のトラックデータ転送における平均サービス時間 : STd-tr＝Tsw＋Ttr＋Tpw.

●各場合の平均応答時間を求める式中で用いられる変数。

I/Oチャネルでのブロックデータ転送の平均応答時間 : STWcp＝Tb＋Tpw＋Wcp.

上下層のMPPM間パスでのブロックデータ転送における1スライスデータ転送の平均応答時間 : STWip＝Ts＋Tpw＋Wip.

ディスク装置のブロックデータ転送における1スライスデータ転送の平均応答時間 : STWd-b＝Tsw＋Ts＋Tpw＋Wd.

ディスク装置のトラックデータ転送における1スライスデータ転送の平均応答時間 : STWd-tr＝Tsw＋Ts＋Tpw＋Wd.

MPPMに書き込まれたブロックデータを読み出す場合、転送ブロックの最初の1スライスデータが書き込まれた時点で、そのブロックデータの読み出しが開始可能となる。

1) ライトバック/ライトバック方式

[各処理要素での平均サービス時間と限界スループット]

1) Channel paths

Scp＝STcp.

Tcp＝(1000×Cp)/Scp.

2) Inter-paths

Sip＝[STip×{R×M1×(1＋D1＋S×D1)＋W×M1×D1}].

Tip＝(1000×Ip)/Sip.

3) Disk drives

R2＝R×M1×(1-D1×S).

W2＝R×M1×D1×(1＋S)＋W×M1×D1.

D2＝R2/(R2＋W2).

Sd＝[STd-tr×R2×M2×(1＋D2)＋STd-b×W2×M2×D2].

Td＝(1000×D)/Sd.

[各場合の平均応答時間]

1) RH ; R×H1×STWcp.

2) WH ; W×H1×STWcp.

3) RM ; R×M1×{S×D1×{H2×[STWip＋STWip＋STWcp]＋M2×{D2×STWd-b＋[STWip＋STWip＋STWcp]}}
＋(1-S×D1)×D1×{H2×{STWip＋M2×{D2×STWd-b＋[STWd-tr＋STWip＋STWcp]}＋H2×[STWip＋STWcp]}
＋M2×{D2×{[STWd-b＋STWip]＋M2×{D2×STWd-b＋[STWd-tr＋STWip＋STWcp]}＋H2×[STWip＋STWcp]}
＋(1-D2)×{STWip＋M2×{D2×STWd-b＋[STWd-tr＋STWip＋STWcp]}＋H2×[STWip＋STWcp]}}}
＋(1-S×D1)(1-D1)×{M2×{D2×STWd-b＋[STWd-tr＋STWip＋STWcp]}＋H2×[STWip＋STWcp]}}}.

4) WM ; W×M1×{D1×{M2×{D2×STWd-b＋[STWip＋STWcp]}＋H2×[STWip＋STWcp]}＋(1-D1)×STWcp}.

2) ライトスルー/ライトスルー方式

[各処理要素での平均サービス時間と限界スループット]

1) Channel paths

Scp=STcp.

Tcp=(1000×Cp)/Scp.

2) Inter-paths

Sip=[STip×{R×M1+W}].

Tip=(1000×Ip)/Sip.

3) Disk drives

R2=R×M1.

W2=W.

Sd=[STd-tr×R2×M2+STd-b×W2].

Td=(1000×D)/Sd.

[各場合の平均応答時間]

1) RH ; R×H1×STWcp.

2) WH ; W×H1×STWcp.

3) RM ; R×M1×{H2×[STWip+STWcp]+M2×[STWd-tr+STWip+STWcp]}.

4) WM ; W×M1×STWcp.

3) ライトスルー/ライトバック方式

[各処理要素での平均サービス時間と限界スループット]

1) Channel paths

Scp=STcp.

Tcp=(1000×Cp)/Scp.

2) Inter-paths

Sip=[STip×{R×M1+W}].

Tip=(1000×Ip)/Sip.

3) Disk drives

R2=R×M1.

W2=W.

D2=R2/(R2+W2).

Sd=[STd-tr×R2×M2×(1+D2)+STd-b×W2×M2×D2].

Td=(1000×D)/Sd.

[各場合の平均応答時間]

1) RH ; R×H1×STWcp.

2) WH ; W×H1×STWcp.

3) RM ; R×M1×{H2×[STWip+STWcp]+M2×{D2×STWd-b+[STWd-tr+STWip+STWcp]}}.

4) WM ; W×M1×STWcp.

4) ライトバック/ライトスルー方式

[各処理要素での平均サービス時間と限界スループット]

1) Channel paths

Scp=STcp.

Tcp=(1000×Cp) / Scp.

2) Inter-paths

Sip=[STip×{R×M1×(1+D1+S×D1)+W×M1×D1}].

Tip=(1000×Ip)/Sip.

3) Disk drives

R2=R×M1×(1-D1×S).

W2=R×M1×D1×(1+S)+W×M1×D1.

Sd=[STd-tr×R2×M2+STd-b×W2].

Td=(1000×D)/Sd.

[各場合の平均応答時間]

1) RH ; R×H1×STWcp.

2) WH ; W×H1×STWcp.

3) RM ; R×M1×{S×D1×[STWip+STWip+STWcp]
+(1-S×D1)×D1×{STWip+STWip+STWcp+M2×STWd-tr}
+(1-S×D1)(1-D1)×{STWip+STWcp+M2×STWd-tr}}.

4) WM ; W×M1×{D1×[STWip+STWcp]+(1-D1)×STWcp}.

付録 II

従来のディスク・サブシステムおよびDIMPにおけるディスクアレイの性能(限界スループット)評価式を以下に挙げる。

各処理要素における限界スループットのうちの最小値がディスク・サブシステム全体の限界スループットとなる。従来のディスク・サブシステムでの処理要素は、入出力チャネルとディスクキャッシュ間の接続パス(C-C paths)、ディスク装置とディスクキャッシュ間の接続パス(D-C paths)、およびディスク装置(Disk drives)である。DIMPの一階層構成での処理要素は、入出力チャネルとMPPM間の接続パス(Channel paths)、およびディスク装置(Disk drives)である。DIMPの場合、ディスク装置とMPPM間の接続パスはディスク装置に専用に設けられている。しかも、このパスでの平均処理時間はディスク装置での平均処理時間に比べて小さく、このパスでの限界スループットはディスク装置での限界スループットに比べて明らかに大きい。したがって、このパスに関する計算は無視する。表1に性能評価用パラメータの一覧を示す。表2に各処理要素での平均サービス時間と限界スループットを表す変数一覧を示す。

表1 性能評価用パラメーター覧

R	I/Oリクエストのリードの割合
W(=1-R)	I/Oリクエストのライトの割合
H	キャッシュのヒット率
Dirty(=W)	ブロックが更新されている確率
Tb	1ブロックの転送時間
Tfb	フェッチブロック(8ブロック固定)の転送時間
Tsw	シークと回転待ちの平均時間
Np	MPPMの入出力ポート数
Tbyte	1バイト(1ポート)の転送時間
Ts(=Np×Tbyte)	1スライスの転送時間
Tpw(=1/2×Ts)	ポートの切り替えによる待ち時間
Cp(=8,64)	入出力チャネルとの接続パス数
Dp(=4,Dall)	ディスク装置との接続パス数
Dc(=8)	全チェックディスク台数
Dd(=64)	全データディスク台数
Dall(=Dc+Dd)	全ディスク台数
Dgroup(=8)	ディスクアレイ・グループ数

表2　各処理要素でのサービス時間と限界スループットを表す変数一覧

Scp	入出力チャネルとの接続パスでの平均サービス時間
Sdp	ディスク装置との接続パスでの平均サービス時間
Sd	ディスク装置での平均サービス時間
Sdc	チェックディスクでの平均サービス時間
Sdd	データディスクでの平均サービス時間
Tcp	入出力チャネルとの接続パスでの限界スループット
Tdp	ディスク装置との接続パスでの限界スループット
Td	ディスク装置での限界スループット
Tdc	チェックディスクでの限界スループット
Tdd	データディスクでの限界スループット

以下で、各処理要素での限界スループットの計算式を示す。ディスクアレイでない場合を便宜上ディスクアレイ方式0と表した。

● 従来のディスク･サブシステムの場合の計算式

[1]　ライトスルー方式

1) C-C paths

ディスクアレイ各方式に関して共通

Scp＝Tb.

Tcp＝(1000×Cp)/Scp.

2) D-C paths

方式0　Sdp＝Tfb×R×M＋Tb×W.

方式1　Sdp＝Tfb×R×M＋Tb×W×2.

方式2,3　Sdp＝Tfb×R×M＋Tb×W.

方式4,5　Sdp＝Tfb×R×M＋Tb×W×4.

以上　Tdp＝(1000×Dp)/Sdp.

3) Disk drives

方式0　Sd＝(Tsw＋Tfb)×R×M＋(Tsw＋Tb)×W.

Td＝(1000×Dd)/Sd.

方式1　Sd＝(Tsw＋Tfb)×R×M＋(Tsw＋Tb)×W×2.

Td＝(1000×Dall)/Sd.

方式2,3　Sd＝(Tsw＋Tfb)×R×M＋(Tsw＋Tb)×W×2.

Td＝(1000×Dgroup)/Sd.

方式4　Sdc＝(Tsw＋Tb)×W×2.

Tdc＝(1000×Dc)/Sdc.

Sdd＝(Tsw＋Tfb)×R×M＋(Tsw＋Tb)×W×2.

Tdd＝(1000×Dd)/Sdd.

Td＝min(Tdc,Tdd).

方式5　Sd＝(Tsw＋Tfb)×R×M＋(Tsw＋Tb)×W×4.

Td＝(1000×Dall)/Sd.

[2]　ライトバック方式

1) C-C paths

ディスクアレイ各方式に関して共通

Scp＝Tb.

Tcp＝(1000×Cp)/Scp.

2) D-C paths

方式0　Sdp＝Tfb×R×M×(Dirty＋1)＋Tb×W×M×Dirty.

方式1　Sdp＝Tfb×R×M×(Dirty×2＋1)＋Tb×W×M×Dirty×2.

方式2,3　Sdp＝Tfb×R×M×(Dirty＋1)＋Tb×W×M×Dirty.

方式4　Sdp＝Tfb×R×M×(Dirty＋1)×2＋Tb×W×M×(Dirty＋1)×2.

方式4.1　Sdp＝Tfb×R×M×(Dirty×3＋1)＋Tb×W×M×Dirty×4.

方式5　Sdp＝Tfb×R×M×(Dirty＋1)×2＋Tb×W×M×(Dirty＋1)×2.

方式5.1　Sdp＝Tfb×R×M×(Dirty×3＋1)＋Tb×W×M×Dirty×4.

以上　Tdp＝(1000×Dp)/Sdp.

3) Disk drives

方式0　Sd＝(Tsw＋Tfb)×R×M×(Dirty＋1)＋(Tsw＋Tb)×W×M×Dirty.

Td＝(1000×Dd)/Sd.

方式1　Sd＝(Tsw＋Tfb)×R×M×(Dirty×2＋1)＋(Tsw＋Tb)×W×M×Dirty×2.

Td＝(1000×Dall)/Sd.

方式2,3　Sd＝(Tsw＋Tb)×R×M×(Dirty＋1)＋(Tsw＋Tb)×W×M×Dirty×2.

Td＝(1000×Dgroup)/Sd.

方式4　Sd＝(Tsw＋Tfb)×R×M×(Dirty＋1)＋(Tsw＋Tb)×W×M×(Dirty＋1).

Tdc＝(1000×Dc)/Sd.

Tdd＝(1000×Dd)/Sd.

Td＝min(Tdc,Tdd).

方式4.1　Sdc＝(Tsw＋Tfb)×R×M×Dirty×2＋(Tsw＋Tb)×W×M×Dirty×2.

Tdc＝(1000×Dc)/Sdc.

Sdd＝(Tsw＋Tfb)×R×M×(Dirty＋1)＋(Tsw＋Tb)×W×M×Dirty×2.

Tdd＝(1000×Dd)/Sdd.

Td＝min(Tdc,Tdd).

方式5　Sd＝(Tsw＋Tfb)×R×M×(Dirty＋1)×2

＋(Tsw＋Tb)×W×M×(Dirty＋1)×2.

Td＝(1000×Dall)/Sd.

方式5.1　Sd＝(Tsw＋Tfb)×R×M×(Dirty×3＋1)＋(Tsw＋Tb)×W×M×Dirty×4.

Td＝(1000×Dall)/Sd.

● DIMPの場合の計算式

[1] ライトスルー方式

1) Channel paths

ディスクアレイ各方式に関して共通

Scp＝Tb.

Tcp＝(1000×Cp)/Scp.

2) Disk drives

方式0 Sd＝(Tsw＋Tfb＋Tpw)×R×M＋(Tsw＋Tb＋Tpw)×W.

Td＝(1000×Dd)/Sd.

方式1 Sd＝(Tsw＋Tfb＋Tpw)×R×M＋(Tsw＋Tb＋Tpw)×W×2.

Td＝(1000×Dall)/Sd.

方式2,3 Sd＝(Tsw＋Tb＋Tpw)×R×M＋(Tsw＋Tb＋Tpw)×W×2.

Td＝(1000×Dgroup)/Sd.

方式4 Sdc＝(Tsw＋Tb＋Tpw)×W×2.

Tdc＝(1000×Dc)/Sdc.

Sdd＝(Tsw＋Tfb＋Tpw)×R×M＋(Tsw＋Tb＋Tpw)×W×2.

Tdd＝(1000×Dd)/Sdd.

Td＝min(Tdc,Tdd).

方式5 Sd＝(Tsw＋Tfb＋Tpw)×R×M＋(Tsw＋Tb＋Tpw)×W×4.

Td＝(1000×Dall)/Sd.

[2] ライトバック方式

1) Channel paths

各ディスクアレイ方式に関して共通

Scp＝Tb.

Tcp＝(1000×Cp)/Scp.

2) Disk drives

方式0 Sd＝(Tsw＋Tfb＋Tpw)×R×M×(Dirty＋1)
＋(Tsw＋Tb＋Tpw)×W×M×Dirty.

Td＝(1000×Dd)/Sd.

方式1 Sd＝(Tsw＋Tfb＋Tpw)×R×M×(Dirty×2＋1)
＋(Tsw＋Tb＋Tpw)×W×M×Dirty×2.

Td＝(1000×Dall)/Sd.

方式2,3 Sd＝(Tsw＋Tb＋Tpw)×R×M×(Dirty＋1)
＋(Tsw＋Tb＋Tpw)×W×M×(Dirty＋1).

Td＝(1000×Dgroup)/Sd.

方式4 Sd＝(Tsw＋Tfb＋Tpw)×R×M×(Dirty＋1)
＋(Tsw＋Tb＋Tpw)×W×M×(Dirty＋1).

Tdc＝(1000×Dc)/Sd.

Tdd＝(1000×Dd)/Sd.

Td＝min(Tdc,Tdd).

方式4.1 Sdc＝(Tsw＋Tfb＋Tpw)×R×M×Dirty×2
＋(Tsw＋Tb＋Tpw)×W×M×Dirty×2.

Tdc＝(1000×Dc)/Sdc.

Sdd＝(Tsw＋Tfb＋Tpw)×R×M×(Dirty＋1)
＋(Tsw＋Tb＋Tpw)×W×M×Dirty×2.

Tdd＝(1000×Dd)/Sdd.

Td＝min(Tdc,Tdd).

方式5 Sd＝(Tsw＋Tfb＋Tpw)×R×M×(Dirty＋1)×2
＋(Tsw＋Tb＋Tpw)×W×M×(Dirty＋1)×2.

Td＝(1000×Dall)/Sd.

方式5.1 Sd＝(Tsw＋Tfb＋Tpw)×R×M×(Dirty×3＋1)
＋(Tsw＋Tb＋Tpw)×W×M×Dirty×4.

Td＝(1000×Dall)/Sd.

R1-10