卒業論文

# マルチポートメモリを用いた
# ハイパーキューブ型マルチプロセッサ
# に関する研究

東北大学工学部情報工学科
丸岡(東)研究室4年

近江　嗣郎

# 目次


## 第１章　序論

１．１　研究の背景・・・・・・・・・・・・・・・・・・・・・・・・・・・１
１．２　研究の目的・・・・・・・・・・・・・・・・・・・・・・・・・・・２
１．３　論文の構成・・・・・・・・・・・・・・・・・・・・・・・・・・・２

## 第２章　ハイパーキューブ

２．１　ハイパーキューブ結合計算機・・・・・・・・・・・・・・・・・・３
２．２　ハイパーキューブ結合計算機の再帰的構成・・・・・・・・・・・４
２．３　ハイパーキューブ経路・・・・・・・・・・・・・・・・・・・・・５

## 第３章　マルチポートメモリ型ハイパーキューブマルチプロセッサ

３．１　システムアーキテクチャ・・・・・・・・・・・・・・・・・・・・６
３．２　通信アーキテクチャ・・・・・・・・・・・・・・・・・・・・・・６

## 第４章　実験

４．１　高速フーリエ変換・・・・・・・・・・・・・・・・・・・・・・・９
４．２　縦形探索・・・・・・・・・・・・・・・・・・・・・・・・・・・１０
４．３　ハードウェア仕様・・・・・・・・・・・・・・・・・・・・・・・１０

4．4　実験方法
4．4．1　デュアルポートメモリを用いたデータ転送・・・・・・12
4．4．2　共有メモリを用いたデータ転送・・・・・・・・・・・13
4．5　実験結果
4．5．1　FFTのデータ数に対する実行時間・・・・・・・・・・17
4．5．2　プロセッサ数に対する実行時間・・・・・・・・・・・17

第5章　結論・・・・・・・・・・・・・・・・・・・・・・・・29

謝辞・・・・・・・・・・・・・・・・・・・・・・・・・・・・30

参考文献・・・・・・・・・・・・・・・・・・・・・・・・・・30

# 第１章　序論

## １．１　研究の背景

高速処理の要求および半導体技術のめざましい発展により、多数のプロセッサを用いた並列処理コンピュータのアルゴリズムやアーキテクチャに関する研究が盛んに行われてきている。科学や工学の数多い分野での各種のアルゴリズムが本質的に並列実行可能であることから、その高速化と性能向上に大きな期待が寄せられている。

マルチプロセッサシステムは、一般にＭＩＭＤ型（Multiple Instruction Stream / Multiple Data Stream）の並列処理方式である。最近は、並列処理計算機のオペレーティングシステム開発の上から、命令とデータの流れによる分類より、並列処理計算機のメモリ形態による分類が一般的になってきており、全プロセッサがメモリを共有する共有メモリ型と、各プロセッサが独自にメモリを持ちメッセージの形でデータを交換する分散メモリ型に分類される。また、プロセッサを共有バスで結合するバス結合型や、各プロセッサを通信リンクを用いて結合するネットワーク型などのマルチプロセッサシステムが開発されている。しかしながら、一般に、マルチプロセッサシステムで採用されているネットワーク通信は、特別な問題に適し、他の問題には適さないということが起こる。理想的には、完全結合やクロスバ結合が最も優れているがコストの面で非常に高くつき現実的でない。また、多くの商業用マルチプロセッサシステムが採用している共有バス結合は、バスバンド幅を大きくし性能を上げているが、1000台を越える超並列マルチプロセッサシステムには適さない。分散メモリ型マルチプロセッサは、その分散された性質のために、大量の並列が可能であるが、効率の良い並列アルゴリズムを設計することはいまだ難しく、膨大な計算能力をフルに活用するためには、プロセッサ間のデータ交換に関する処理がオーバヘッドとなってしまう。この方式を用いたハイパーキューブ型マルチプロセッサシステムも商業機として採用されているが、それらのシステムでは、プロセッサ間通信に従来のI/Oイン

ターフェースを用いたハンドシェイクが採用されているため、メッセージ交換でアプリケーションプログラムを実行する場合オーバヘッドが非常に大きくなり性能向上は得られない。

## 1．2　研究の目的

このように、各種並列アルゴリズムならびにマルチプロセッサシステムの研究が進むにつれて、高並列になるほど計算量複雑度より通信複雑度が並列処理の高速化を支配することが分かってきた。従って、並列処理を効率よくするためには、プロセッサ間通信のオーバヘッドをかなり小さくしなければならないことが必要条件となる。

本研究では、マルチプロセッサシステムの性能を決定するプロセッサ間通信を高速にするために、プロセッサ間結合ネットワークとして最も優れていると考えられているハイパーキューブ結合を採用して、マルチポートメモリを用いた通信アーキテクチャを提案し、その性能をシミュレーションによって評価する。

## 1．3　論文の構成

第1章は序論である。第2章ではハイパーキューブ結合計算機の構成やその特徴について簡単に述べる。第3章では、提案されるマルチポートメモリ型システムの構成・特徴をシステムアーキテクチャと通信アーキテクチャの点から述べる。第4章は実験についてである。システムのシミュレーションに用いた高速フーリエ変換と縦形探索について簡単に述べた後、実際のシステムの仕様とシミュレーションに用いたパラメータについて述べ、実験方法とその実験結果を示す。第5章は結論である。

# 第２章　ハイパーキューブ

## ２．１　ハイパーキューブ結合計算機

ハイパーキューブ結合は、数あるプロセッサ間結合方式の中で最も有力なものの一つである。$2^m$台のプロセッサからなるｍ次元ハイパーキューブ結合計算機は次のように定義される。$2^m$台のプロセッサに０，１，２，…，$2^m-1$なる自然数アドレスをつける。プロセッサ$P_i$（$0 \leqq i \leqq 2^m-1$）のアドレスｉを２進表現したものを

$$(i_{m-1}\ i_{m-2}\ \cdots\ i_2\ i_1\ i_0)_2$$

とする。右端をl.s.b.とする。プロセッサ・アドレスと二つのプロセッサを結合する通信リンクとの間には次の関係がある。

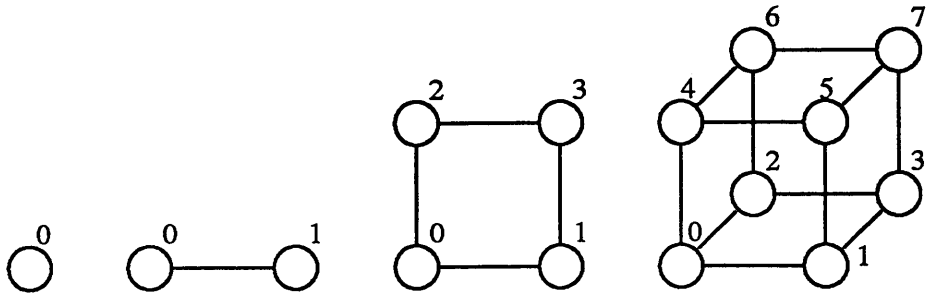
（１）各プロセッサ$P_i$（$0 \leqq i \leqq 2^m-1$）は、ｍ本の通信リンクを持つ。

（２）Ｐｉのもつｍ本の通信リンクは、次のプロセッサ・アドレスをもつｍ個のプロセッサと接続される。すなわち、

$$(i_{m-1}\ i_{m-2}\ \cdots\ i_2\ i_1\ \overline{i_0})_2$$
$$(i_{m-1}\ i_{m-2}\ \cdots\ i_2\ \overline{i_1}\ i_0)_2$$
$$(i_{m-1}\ i_{m-2}\ \cdots\ \overline{i_2}\ i_1\ i_0)_2$$
$$\vdots$$
$$(\overline{i_{m-1}}\ i_{m-2}\ \cdots\ i_2\ i_1\ i_0)_2$$

なるｍ個のプロセッサである。ここで$\overline{i_k}$（$0 \leqq k \leqq m-1$）は$i_k$のビット反転を意味する。図１に０，１，２および３次元のハイパーキューブ結合並列計算機の構成図を示す。○はプロセッサを、その横の数字はプロセッサ・アドレスを、○を接続する線は２方向通信リンクを意味する。

直接に通信リンクで接続されているプロセッサを、隣接プロセッサと呼ぶ。ｍ次元ハイパーキューブ結合計算機においては、各プロセッサはｍ（$=\log_2 n$）個の隣接プロセッサをもつ。したがってリンク総数は全体で（$n \log n$）／２である。




(a) 0 次元　(b) 1 次元　(c) 2 次元　(d) 3 次元

図1　0，1，2，3次元ハイパーキューブ結合計算機

隣接プロセッサ数は、メッシュ結合が定数個であるのに対して、ハイパーキューブ結合では、プロセッサ数の増加にともない、対数関数の割合で増えてゆく。また、各プロセッサは局所メモリをもち、スケーラビリティを向上させるため分散型メモリアーキテクチャを採用するのが一般的である。

## 2．2　ハイパーキューブ結合計算機の再帰的構成

m次元ハイパーキューブ結合計算機は、次のように、m－1次元ハイパーキューブ結合計算機から再帰的に構成できる。

（1）m－1次元ハイパーキューブ結合計算機$M_0$，$M_1$を各1個ずつ用意する。

（2）$M_0$と$M_1$において同一アドレスをもつプロセッサどうしを合計$2^{m-1}$本の通信リンクで接続する。

（3）$M_0$および$M_1$を構成するプロセッサの最上位2進アドレスとして、それぞれ0および1を追加し、mビットアドレスとする。

上記の定義は、0次元の1台のプロセッサから出発して、プロセッサ台数を倍々に増やしながら、所定の次元のハイパーキューブ結合計算機を再帰的に構成できることを意味する。この再帰的構成法やその逆の縮退プロセスは、プロセッサ$P_0$からすべてのプロセッサにデータをブロードキャストしたり、各プロセッサのもつデータに半群演算を施して、その結果を$P_0$に集積する場合などに役立つ。

## 2．3　ハイパーキューブ経路

二つのプロセッサ$P_i$，$P_j$間のハミング距離H（i，j）とは、プロセッサ・アドレスi，jの2進表現で各桁における相違するビット数の合計である。たとえば$P_{0001}$と$P_{1110}$間のハミング距離は4である。ハイパーキューブ結合では、H（i，j）＝1なるプロセッサ$P_i$，$P_j$に限り、直接にリンクで結合されるといえる。

直接にリンクで接続されないプロセッサ間の通信は、これらを結ぶパス上のプロセッサを経由して、メッセージを伝達することによって行われる。任意のプロセッサ$P_i$，$P_j$を結ぶ最短パスの長さはH（i，j）である。これは、i，jの2進表現において相違するビットの値を、1回につき1ビット反転させたアドレスをもつプロセッサをたどることで得られる。たとえば、$P_{0000}$から$P_{1111}$に至る長さ4のパスは、$P_{0000}$→$P_{1000}$→$P_{1100}$→$P_{1110}$→$P_{1111}$という具合である（図2）。

したがって、最も遠く離れているプロセッサ間の距離はlognであり、平均距離は(logn)／2である。

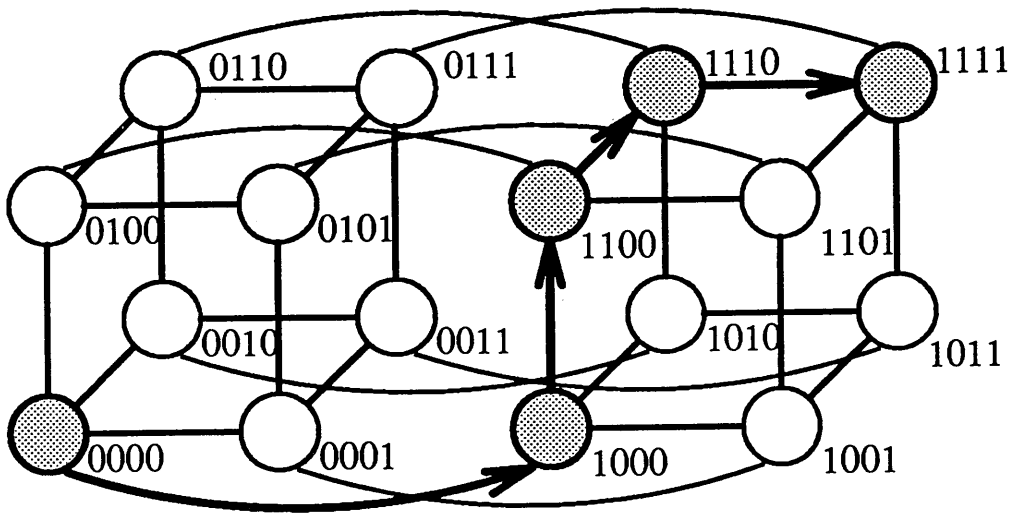



図2　4次元ハイパーキューブにおける$P_{0000}$から$P_{1111}$へ至る経路

# 第３章　マルチポートメモリ型ハイパーキューブマルチプロセッサ

## ３．１　システムアーキテクチャ

マルチプロセッサシステムの処理能力はプロセッサ間のデータ転送能力に深く依存する。そこで、プロセッサ間の高速通信が期待できるハイパーキューブ結合を基本とし、更にデータ転送と処理が独立に行える新しいシステムが提案された［１］。本システムはプロセッサ間の１対１通信、１対多通信を高速に行えるようにマルチポートメモリをもちいたデータ転送ハードウェアを備えている。またソフトウェアのポーティングの容易性も考慮した共有メモリ構造にもなっており、高速メッセージ交換、データ通信が行えるシステムに設計されている。各プロセッサは、MPU（Micro Processing Unit）、ローカルメモリ、マルチポートメモリ、データ転送用ハードウェア、共通バスからなる。ハイパーキューブネットワークにはデータ転送用のスイッチがあり、MPUをネットワークから切り放す役割をする。データ転送にはマルチポートメモリと独立したデータ転送用バスが使用され、MPUは通信リンクが確立されればデータの始まりとブロック数のみ指定すれば自動的にデータ転送が行われる。マルチプロセッサのアーキテクチャはハイパーキューブ結合方式のネットワーク構成でＭＩＭＤ型の処理を行う。マルチプロセッサシステムを構成する各ユニットはMPUを有しそれぞれにメモリやI/Oを備える独立したプロセッサシステムである。ただし２次記憶装置やユーザーインターフェースは用意せず、ホストシステムのサービスを利用する。

## ３．２　通信アーキテクチャ

プロセッサ間通信はマルチポートメモリを用いて実現される。データ転送はMPUのバスとは独立したバス（以下リンクと記述）を用いてブロック単位で行われ、高速データ転送を実現する。転送はMPUの監視を必要とせずMPUバス

も使用しないので、MPUの利用効率を上げることができる。リンクは基本的にn次元ハイパーキューブ結合を形成するが、隣接するユニット間の通信だけでなく任意のユニット間の通信も直接行える。この場合、転送経路上のユニットはリンクを使用することはできないが、それ以外の制限は受けない。従って、プロセッサ間のデータ転送は1対1のデータ転送の他に、ブロードキャスト通信、1対多のデータ通信が可能である。マルチポートメモリの一つであるデュアルポートメモリを用いたプロセッサ間データ転送ハードウェアの構成概念図を図3に示す。また、マルチプロセッサシステムはVME BUSのバスマスターとしての機能を持ち、VME BUS上に接続されたメモリボードを介してホストシステムからのデータ転送、制御管理、共有I/Oのアクセスを行うことができる。

ハイパーキューブ結合におけるプロセッサ間通信はシステム内で5つの階層構造（物理層、データリンク層、ネットワーク層、トランスポート層、セッション層）に分けて考えることができる。物理層は物理的通信技術を提供し、実際の通信ハードウェアに相当する。データリンク層は、プロセッサノード間の通信技術を提供する。ネットワーク層では、経路制御や中継を行う。ハイパーキューブ結合の最短経路選択は簡単なビット制御のアルゴリズムによって実現可能である。ただし、故障ノードの回避、トラヒックによる経路の選択も考慮にいれたアルゴリズムでなければならない。トランスポート層では、指定アドレスのプロセッサノード間でのデータ転送を行う。プロセッサ間データ転送には、メッセージ通信とリンクモードの2つのモードを用意している。メッセージ通信モードは、パケット通信を基本としており、プロセッサ間通信を行う。一方、リンクモードは、プロセッサ間に直接物理リンクを生成し、大量のデータを高速転送する（図4）。セッション層では、プロセスが利用できる全体的通信手段をシステムコールとしてユーザレベルに対応する上位層に提供する。内部では、エラー回復、順序制御、流量制御、データ転送方式の選択などを行う。

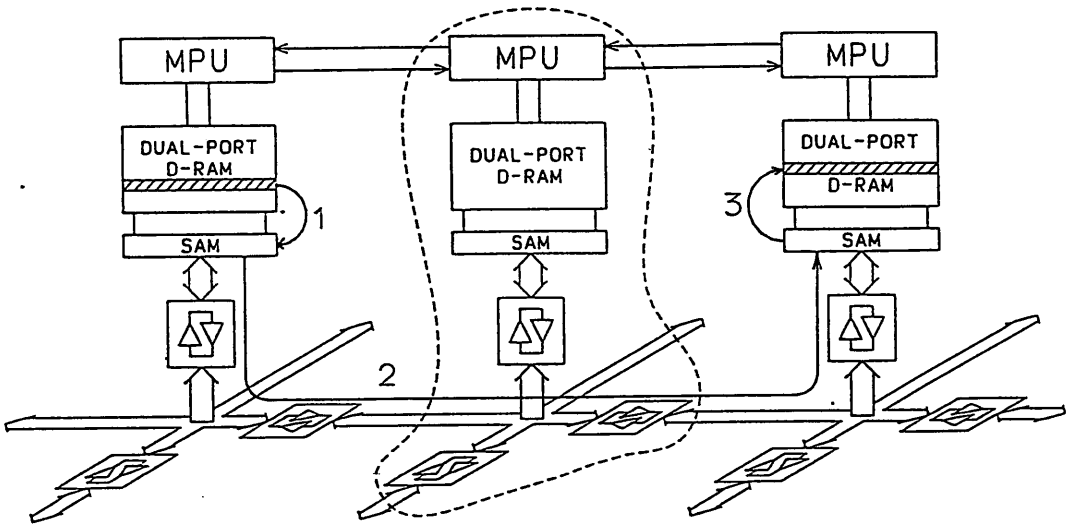



図３　プロセッサ間データ転送ハードウェアの構成

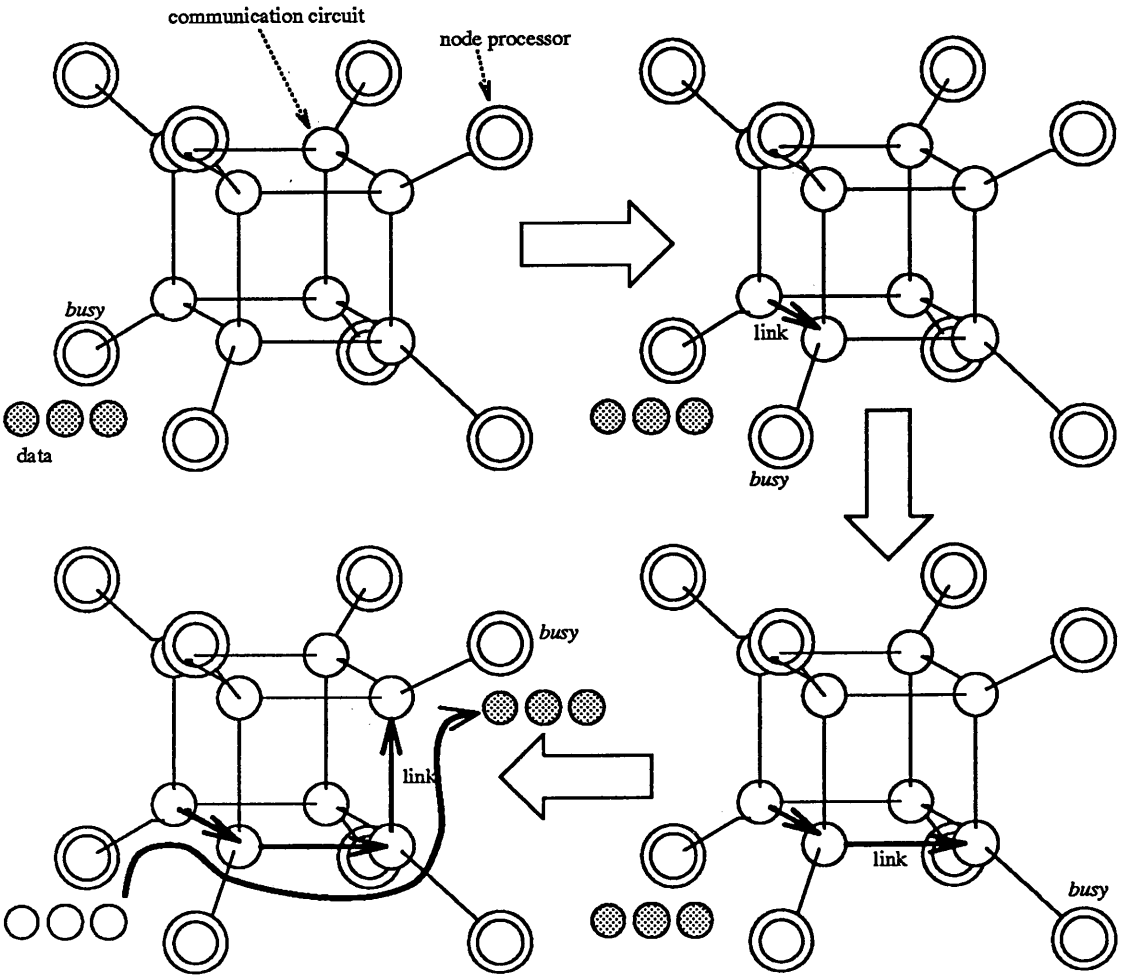



図４　ハイパーキューブ結合のリンク通信方式

# 第４章　実験

## ４．１　高速フーリエ変換

マルチポートメモリ型ハイパーキューブシステムの動作時間の比較シミュレーションとして、高速フーリエ変換（FFT）を採用する。

ここで、Ｎ個の複素入力データ列を扱うとすれば、離散フーリエ変換では実数演算でおよそ８$N^2$の計算量が必要である。FFTの逐次処理の場合には、一つのバタフライ演算はF+WG,F-WGの形式をしているので、合計１０実数演算が必要である。バタフライ演算は１ステージ当たりN/2であり、ステージ数はlogNであるので、全体として、実数演算で５NlogN演算必要である。１実数時間を$\alpha$とすると、逐次処理の場合、処理時間は

$$5N\alpha\log N$$

となる。

並列処理の場合、Ｐ個のプロセッサがネットワークに結合されており、各プロセッサに入力データ列の内の連続するN/P個が格納されているとする。Ｎ点FFTは最初のlogＰ段までの処理はプロセッサ間通信が必要であり、後半のlog(N/P)段の処理は各プロセッサ内で逐次処理される。logＰ段までの演算では、各プロセッサはF+WGまたはF-WGのいずれかのバタフライ演算を実行し、これに８実数演算が必要となる。各プロセッサで１ステージ当たりN/P個の演算をし、ステージ数はlogＰであるので、logＰ段までの演算時間は

$$(8N\alpha/P)\log P \qquad \cdots\cdots(4.1)$$

後半のlog(N/P)ステージに対しては、各プロセッサ内で逐次処理されるので、

$$(5N\alpha/P)\log(N/P) \qquad \cdots\cdots(4.2)$$

したがって、並列処理では、式(4.1)，(4.2)及びプロセッサ間でのデータの転送時間の和が処理時間となる。

シミュレーションでは、入力データ数及びプロセッサ数による処理時間をデータの通信方式の観点から評価を行うことにする。

## 4．2　縦形探索（depth-first search）

ハイパーキューブ結合の最短経路選択は簡単なビット制御のアルゴリズムによって実現可能であるが、故障ノードの回避、トラヒックによる経路の選択も考慮にいれたものでなければならない。そこで、このアルゴリズムとして縦形探索（DFS）を採用した。これは、グラフの点を漏れなくたどる方法の一つで、一つの点から出発して進めるだけ進み、行き止まりになったら後戻りして新しい道を探すアルゴリズムである。しかしながら、ハミング距離が2以上である場合にはこのアルゴリズムが必要であるが、本論文で取り上げるFFTについて言えば、データを通信するべきノード間の距離は全て1であるので、経路選択の必要がないことになる。また、本シミュレーションでは、ノードやリンクの故障はないものとしている。

## 4．3　ハードウェア仕様

試作マルチプロセッサシステムのハードウェアの仕様を表1に示す。プロセッサユニットはCPUとしてMC68000 MPUを採用している。各プロセッサはメモリやI/Oを有する独立したプロセッサシステムである。コプロセッサであるMC68882は浮動小数点演算プロセッサであり16MHzで動作する。

プロセッサ間通信はマルチポートメモリの1つであるデュアルポートDRAMのシリアル入出力機能を用いて実現される。データ転送はCPUのバスとは独立したリンクを用いて512bytesのブロック単位で行われ、その転送速度は4M bytes/secである。転送はMPUの監視を必要とせず、MPUバスも使用しないので転送終了時発生する割り込みを受け付けるまでの間MPUは処理を継続することができる。また、転送経路上のユニットはリンクを使用することができないが、それ以外の制限は受けない。

また、ワークエリアをローカルメモリとし、各メモリへのアクセス時間を500nsec/1word、共有メモリとのデータ転送速度を2Mbytes/secとする。

| | | |
|---|---|---|
| CPU | MC68000-8 MPU | 8MHzで駆動。メインメモリ・通信用メモリなどのほとんどのデバイスのアクセスはno waitで行われる。 |
| 周辺LSI | MC68230-8 PI/T | 通信回路のコントロールやタイマ割り込みの発生を行う。 |
| | MC68882-16 FPC | 16MHzで駆動。浮動小数点演算を行う。 |
| | SBC68172 BUSCON | バス調停などのVMEバスコントロールを行う。 |
| メモリ | ROM 128KB | IPLを格納する。DRAM 512Kbytesを主記憶として用いる。 |
| | DP-DRAM 128KB | 通信用メモリとして用いる。 |
| | SRAM 16KB | バッテリバックアップされ、メモリスイッチ用に用いる。 |
| 電源 | 5V±5% | |

表1　　プロセッサユニットのハードウェア仕様

## 4．4　実験方法

システムシミュレーションとして、FFTのバタフライ演算部の処理時間に関する比較評価を行う。ハイパーキューブ上の処理時間を考える場合には、各プロセッサ内で行われる演算時間とプロセッサ間でのデータ転送時間の和を考慮する必要がある。

プロセッサ内の演算時間は、バタフライ演算部をMC68000に関するアセンブラコードに書換え、それらのクロックを合計することで得られる。また、プロセッサ間のデータ転送時間は、転送方法によりその獲得方法が異なる。本シミュレーションでは以下の二つの転送方法によって転送時間を求める。

### 4．4．1　デュアルポートメモリを用いたデータ転送

プロセッサ間通信にDP-DRAMを用いる場合、一般に、転送されるデータのサイズやデータ列の長さにより、DP-DRAMとリンクの上でのデータの遷移の状態は変わってくる。その模式図を図5，6，7，8に示す。それぞれの図におけるrequestとは、各プロセッサ内で前段階の演算終了時に発生するデータ要求を意味する。採用したFFTのバタフライ演算部は次のようになっている。

```
dx = s*yy + c*xx;
dy = c*yy - s*xx;
xx = xxx - dx; xxx += dx;
yy = yyy - dy; yyy += dy;
```

なお、x,yに関する式はそれぞれ実部、虚部の演算であり、sおよびcは三角関数表の値である。例えば図5では、

①CPU[A]でdxが計算される。この時、CPU[B]ではdxのrequestが出され、データ待ちとなる。

②dxが計算されるとCPU[B]のrequestにより、まずDP-DRAMにdxが送られる。この時、CPU[A]でxxxのrequestが出されるので、CPU[B]はxxxを送りはじめる。

③CPU[A]がDP-DRAMへのデータ転送を終了すると、直ちにリンクへの転送が開始され、CPU[A]はデータ待ちとなる。一方、CPU[B]がDP-DRAMへのデータ転送を終えたとしても、リンクは使用中であるので待機することになる。

④CPU[A]側のリンクへのデータ転送が終わるとそれがCPU[B]側のDP-DRAMへ送られる。同時に、CPU[B]側のDP-DRAMからリンクへxxxが転送される。

⑤それぞれのデータが各CPUへ転送され、演算処理される。

以下、それぞれのデータサイズやデータ列の長さ、また、その転送時間の差を考慮に入れれば、図６，７，８に関しても同様にして得ることができる。

一般的には、以上のことが言える。しかしながら、FFTの場合には、dx,xx,xxx,yy,yyyの各データサイズが同じこと、転送されるデータブロック長が決められていることを考えれば、その模式図は図９のように統一される。なお、データ転送は512bytesのブロック単位で行われ、その転送速度は4Mbytes/sec、各メモリへのアクセス速度は500nsec/1wordである。また、１データ長は2bytesである。

### ４．４．２　共有メモリを用いたデータ転送

データ転送に共有メモリを用いる場合には、複数のプロセッサが同時に共有メモリをアクセスすることはできない。アクセスするには、あるプロセッサ間通信が終了してからということになるため、その様子はパイプライン的になる。この模式図を図10に示す。なお、共有メモリとのデータ転送速度は2Mbytes/secである。また、１データ長は2bytesである。

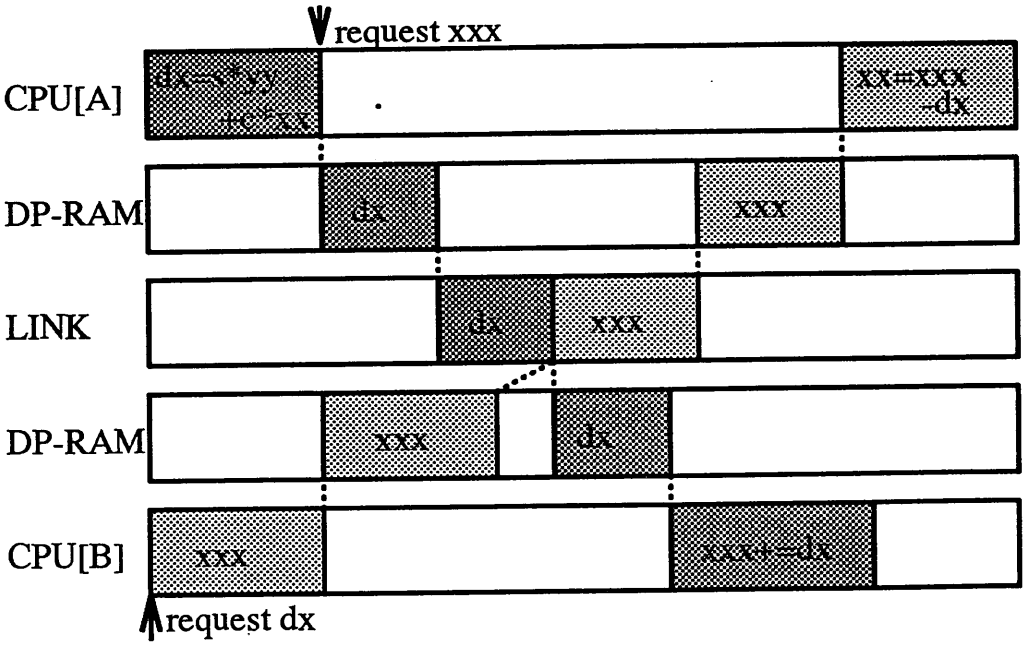



図５　デュアルポートメモリを用いたデータ転送（１）

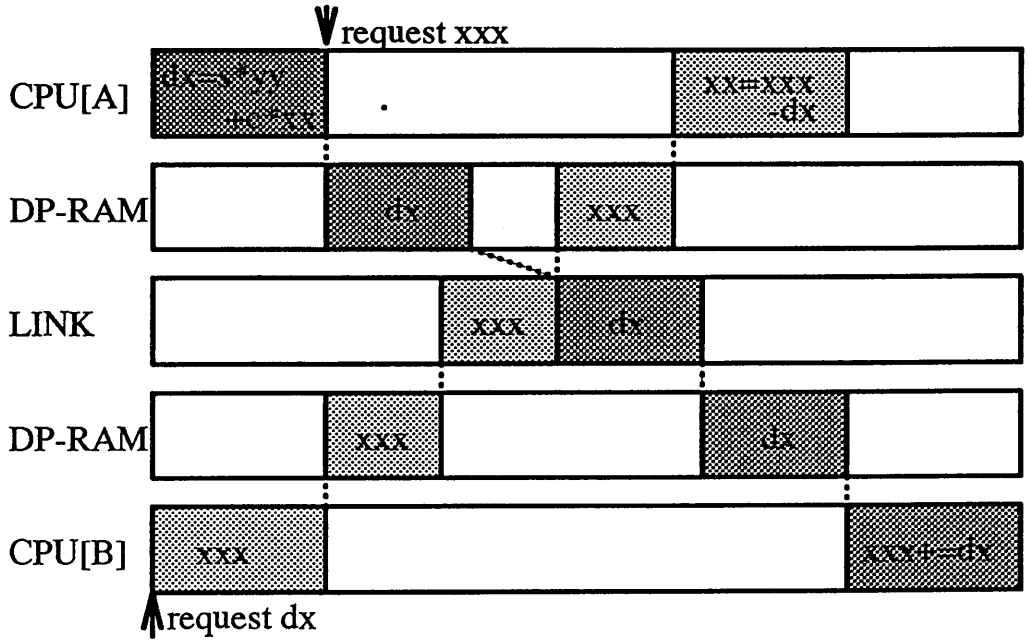



図６　デュアルポートメモリを用いたデータ転送（２）

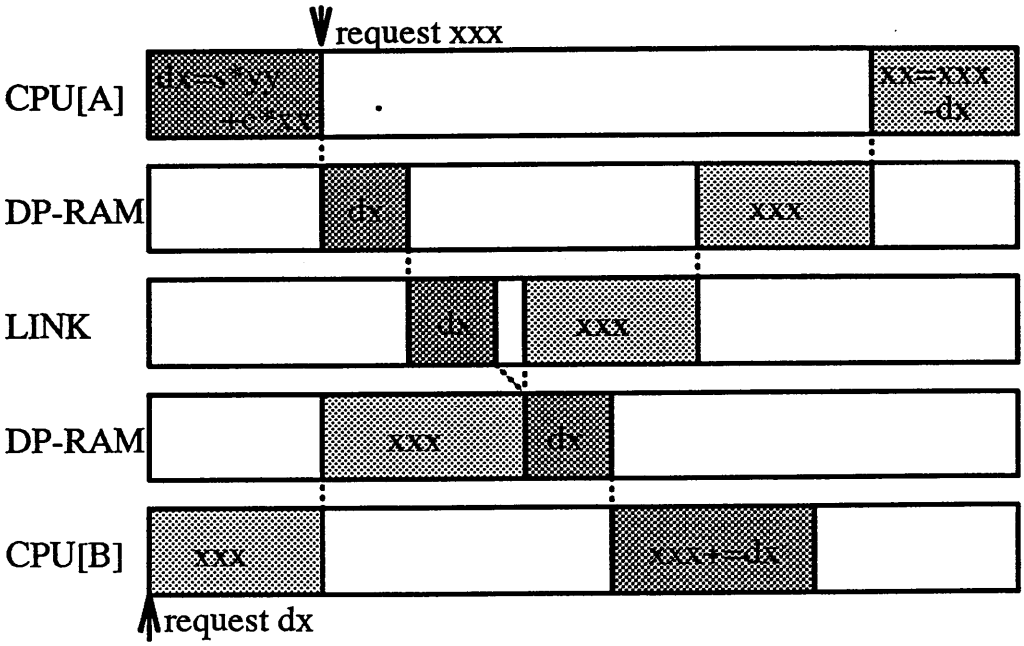



図７　デュアルポートメモリを用いたデータ転送（３）

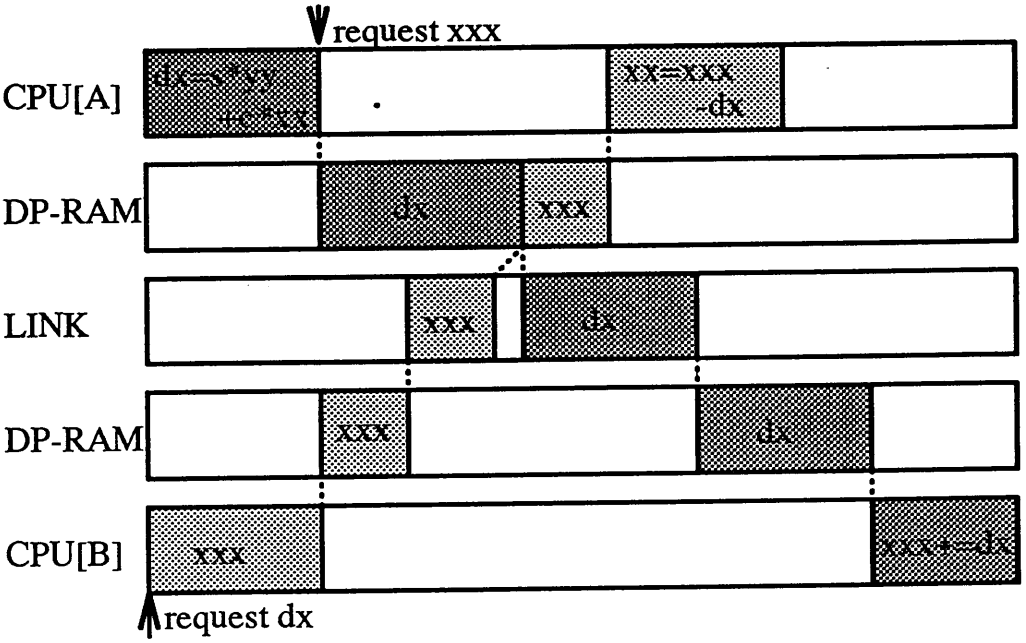



図８　デュアルポートメモリを用いたデータ転送（４）

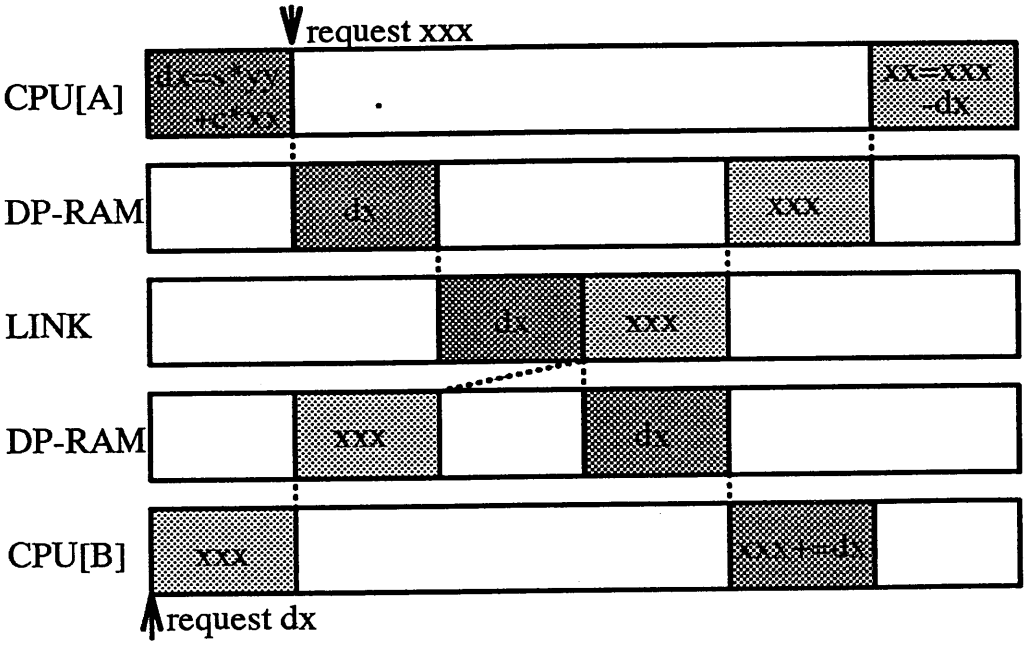



図９　デュアルポートメモリを用いたFFTのデータ転送

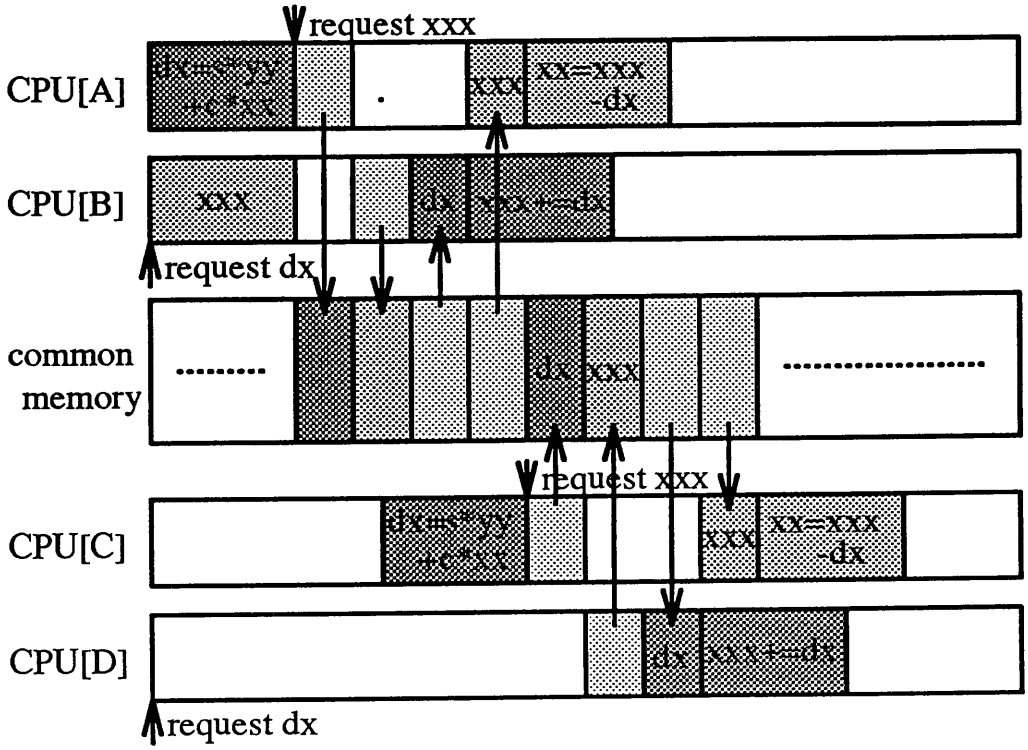



図１０　共有メモリを用いたデータ転送

## ４．５　実験結果

### ４．５．１　FFTのデータ数に対する実行時間

DP-DRAMを用いたときの実行時間を図11に示す。プロセッサ数が４以上の場合、結果が線形ではないことが分かる。これは、データ転送の際、512bytesのブロック単位で行われることがボトルネックとなっているためと考えられる。しかしながら、プロセッサ数１，２の場合を除けば、プロセッサ数の増加によって、かなりの速度向上がみられる。これに対して、共有メモリを用いた場合（図12）では、16以上のプロセッサ数では、ほとんど速度向上が得られない。

図13から図19には、同プロセッサ数での二つの通信方式による処理時間の差を示した。プロセッサ数４と８では前述のボトルネックによって、共有メモリを用いたものの方が良い結果となっているが、それ以上では、ある範囲においてDP-DRAMを用いた方式の方がかなりの速度向上を得ている。

### ４．５．２　プロセッサ数に対する実行時間

DP-DRAMを用いたときの結果を図20に示す。FFTのデータ数が512個以上の場合、実行時間は単調減少ではなく、グラフから、前述のボトルネックがかなりの悪影響であることが理解できる。しかしながら、プロセッサ数１，２の場合を除けば、プロセッサ数を増加させることによって、かなりの速度向上がみられる。これに対して、共有メモリを用いた場合（図21）は、FFTのデータ数が512個以上では、ほとんど速度向上が得られない。両図を比較すると、DP-DRAM型の方は、速度向上を得るためには、データ数が多いほどプロセッサ数もまた多くしなければならないことが分かる。

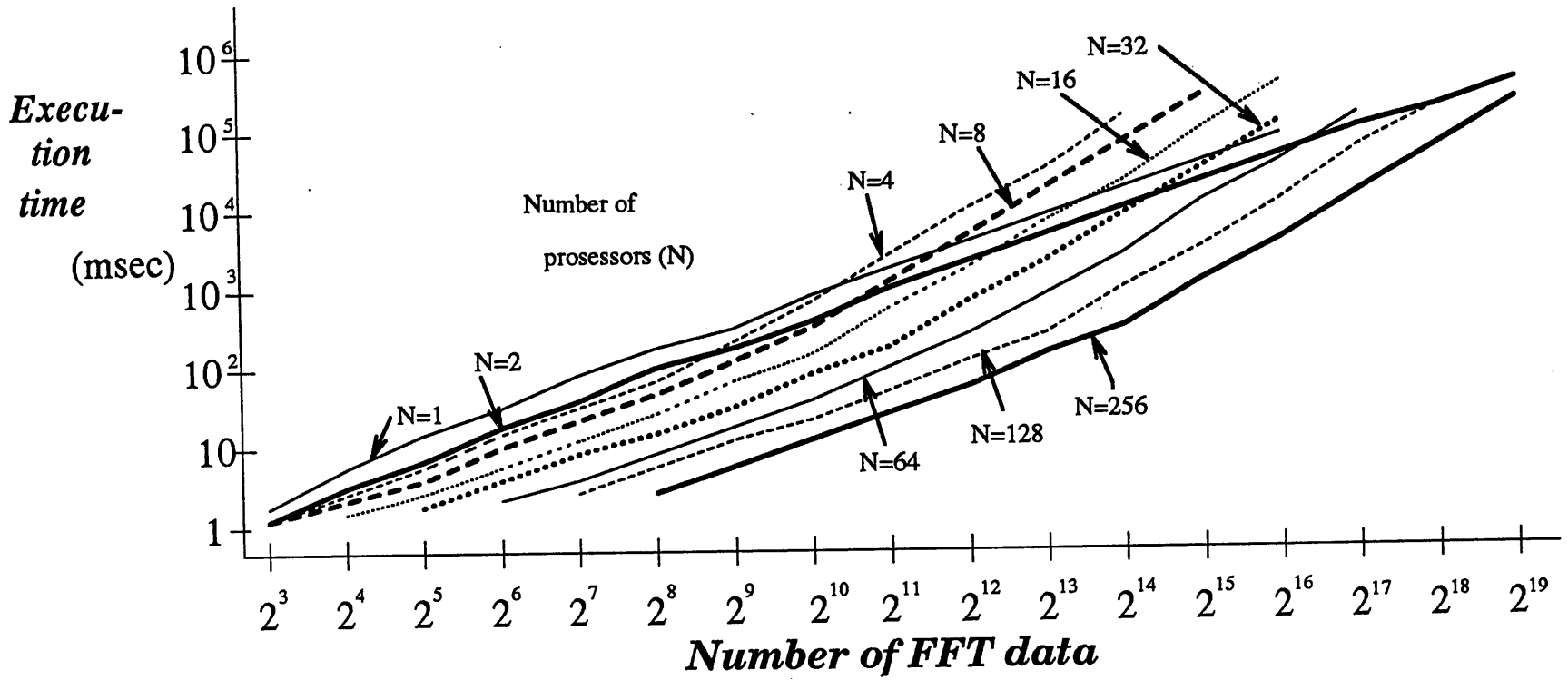



図１１ Effects of FFT data size (using DP-RAM)

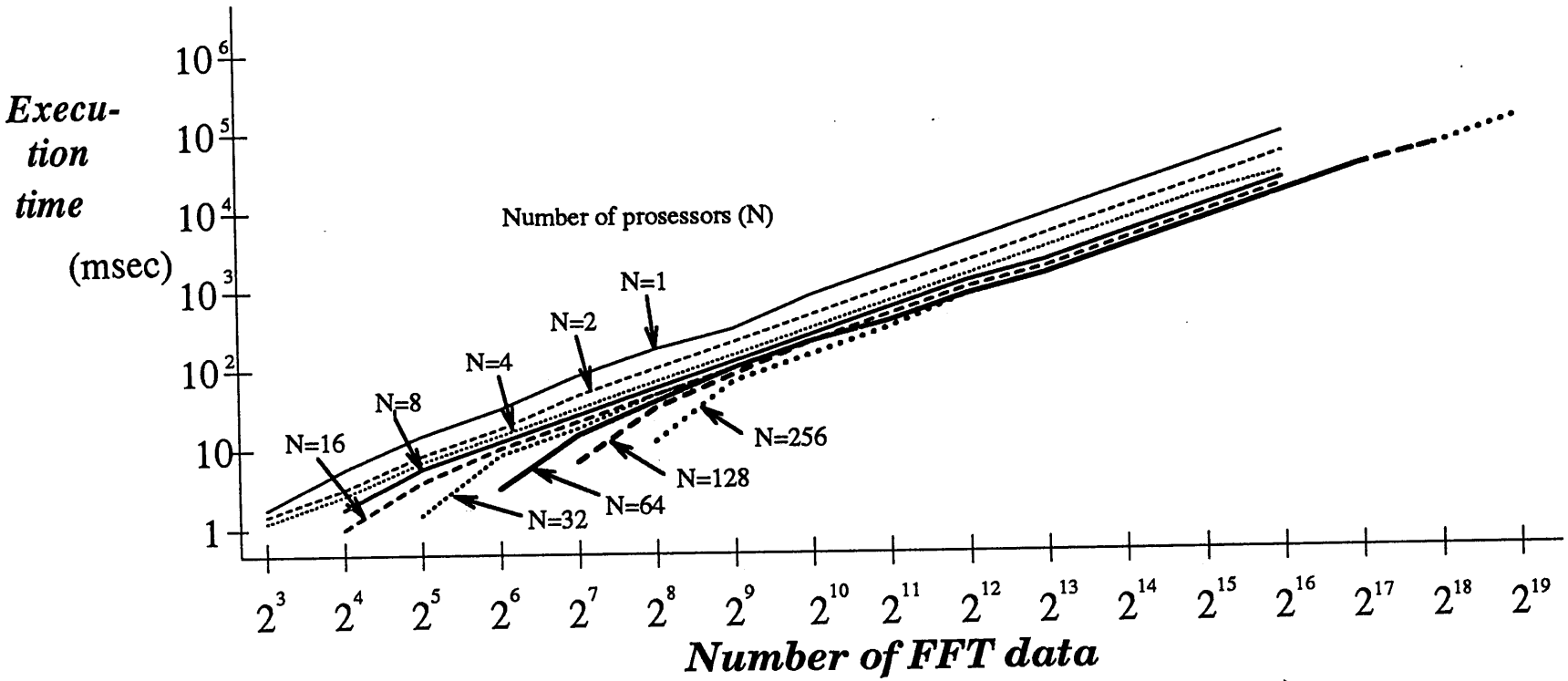



図１２ **Effects of FFT data size (using common memory)**

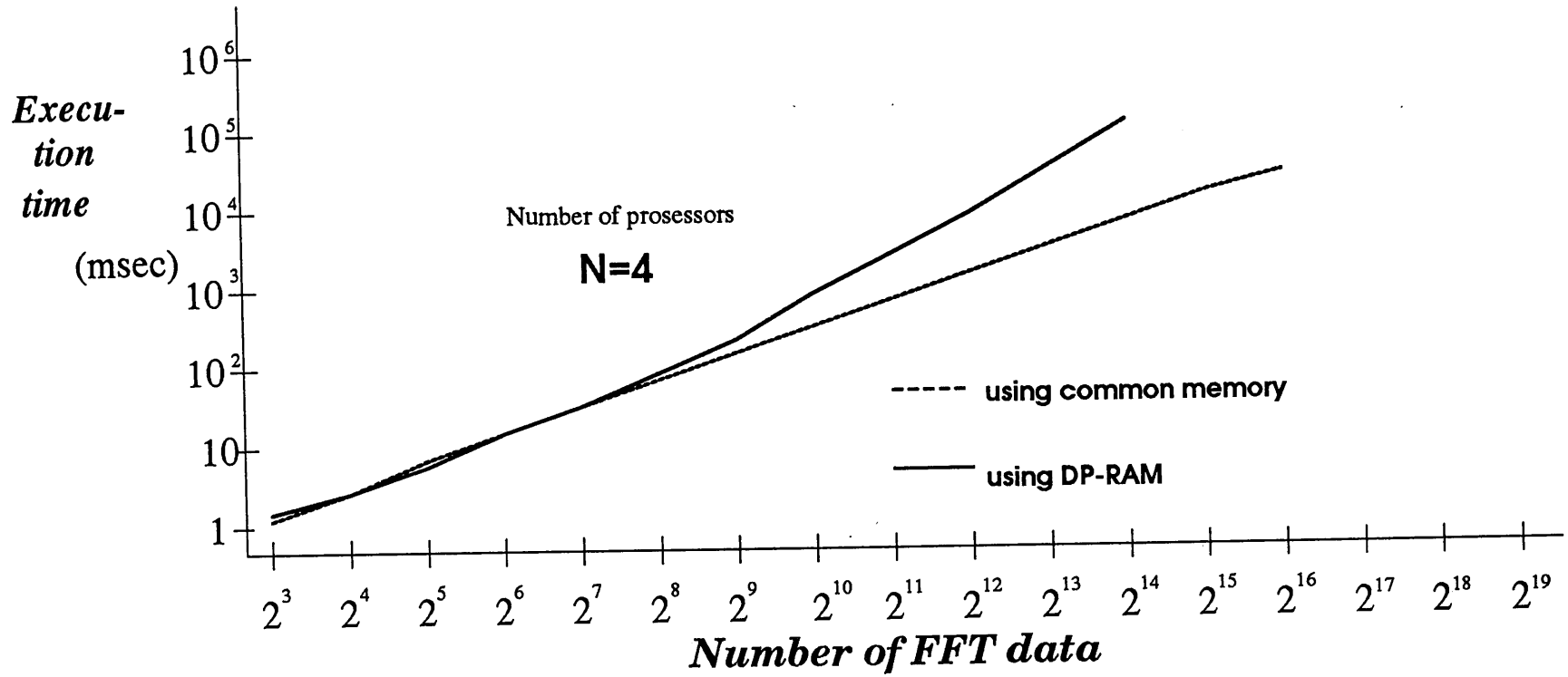



図１３ Comparison of execution types

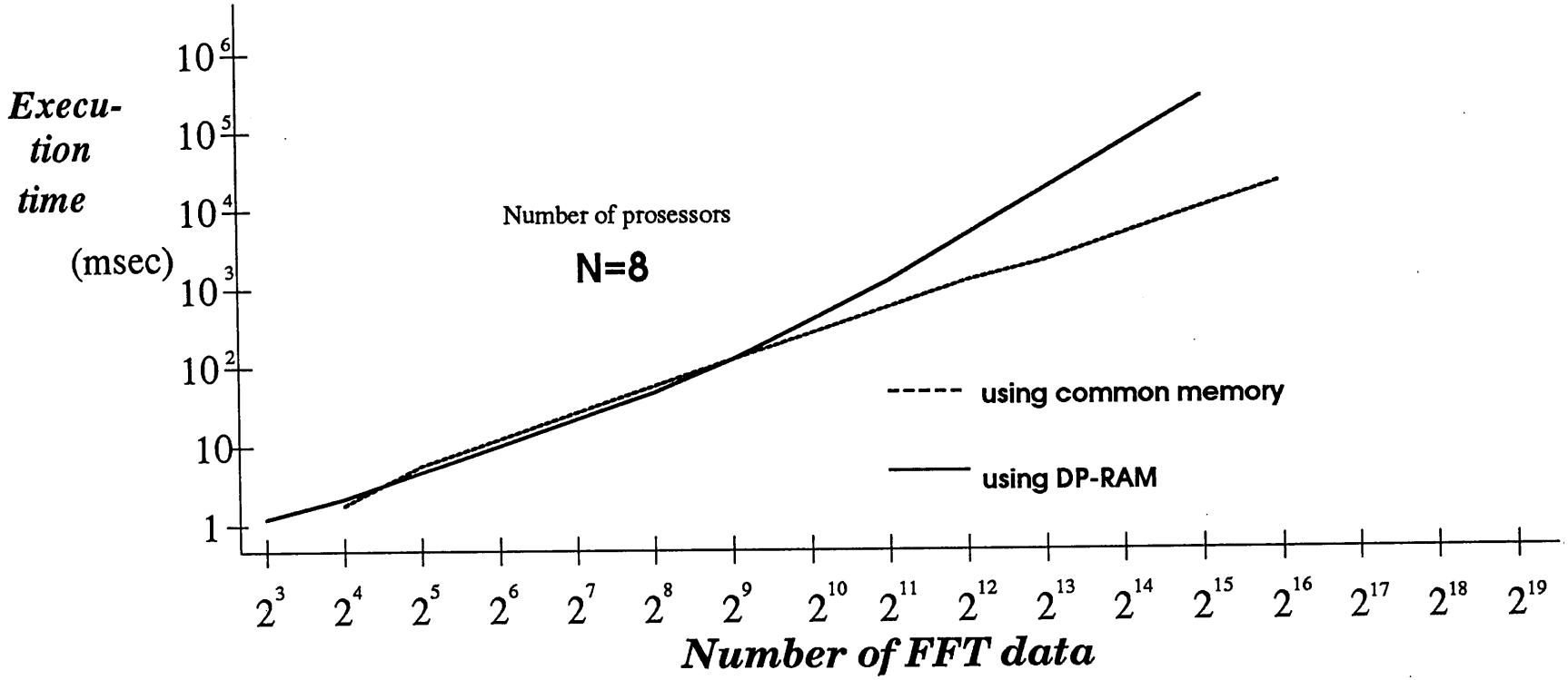



図１４ Comparison of execution types

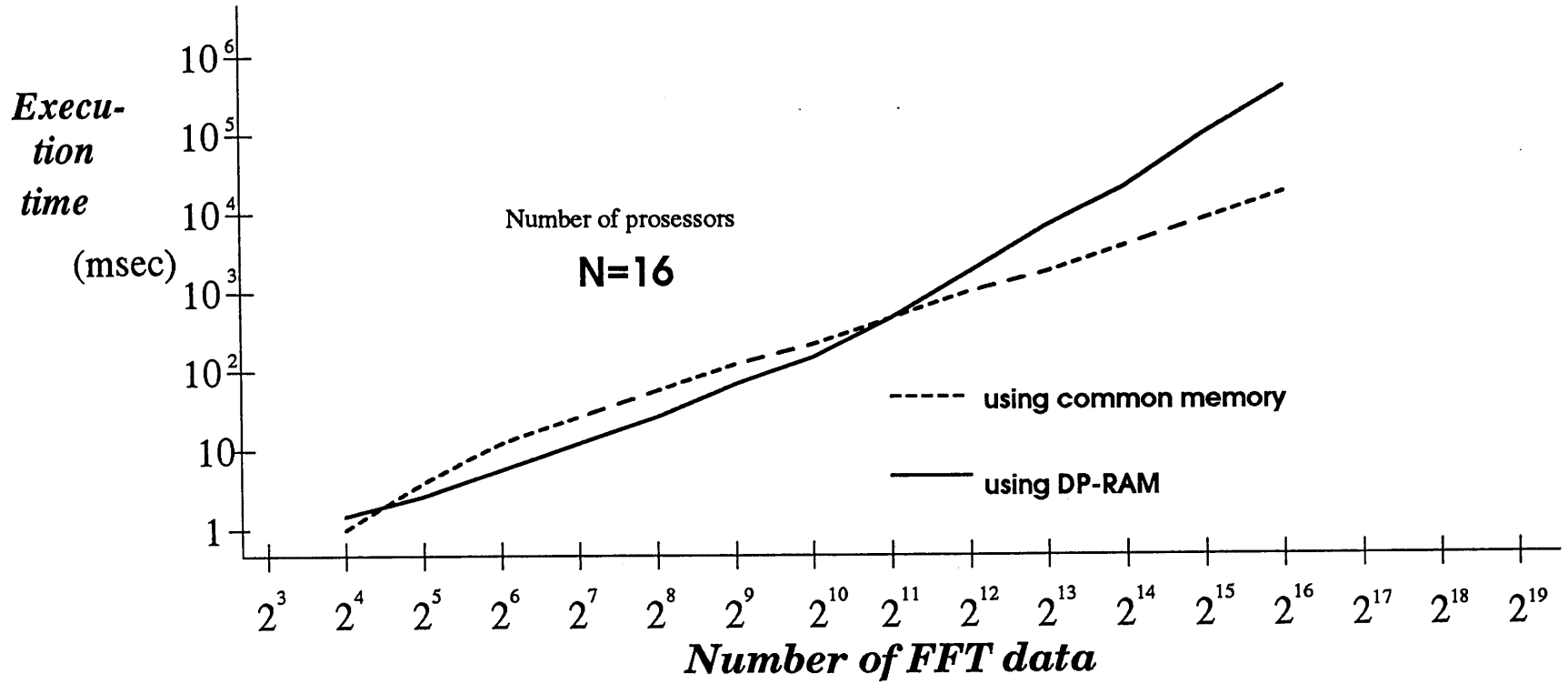



図１５ Comparison of execution types

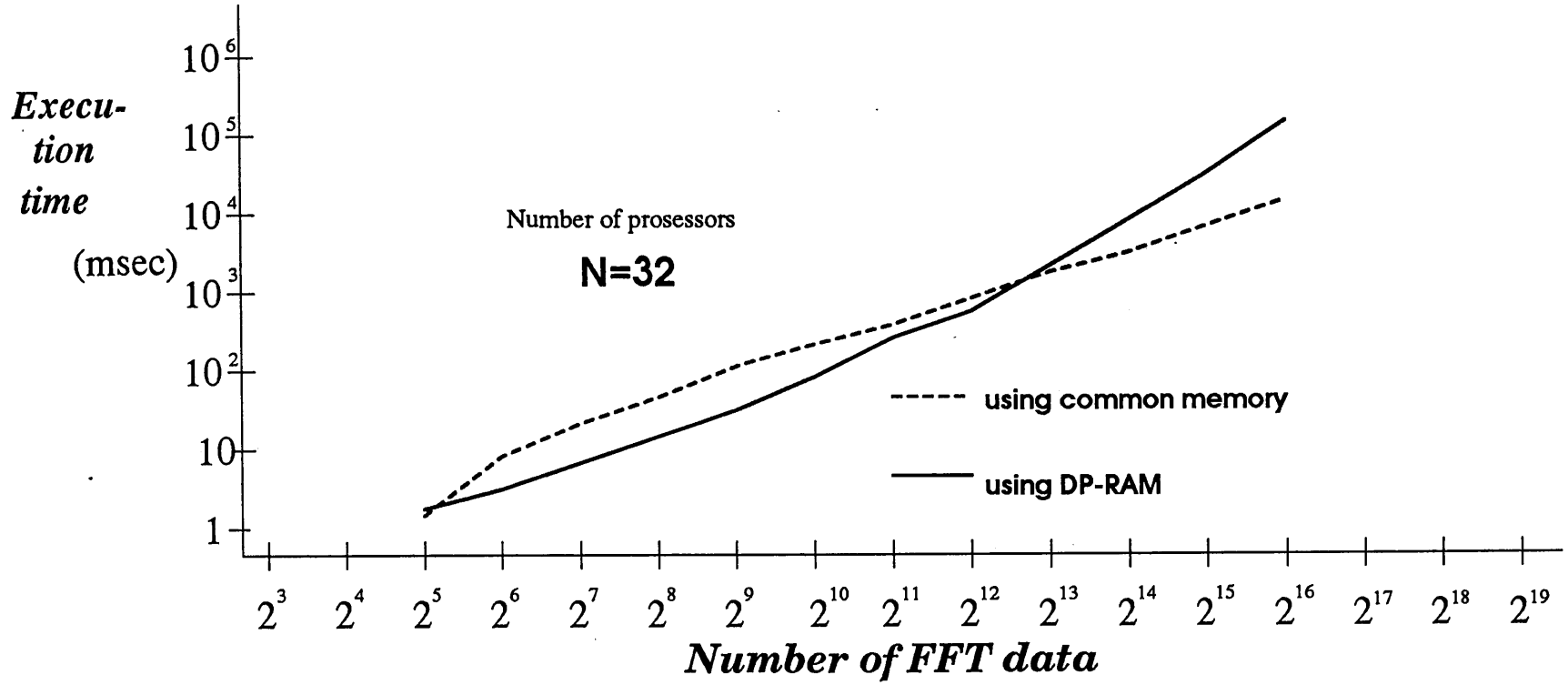



図１６ Comparison of execution types

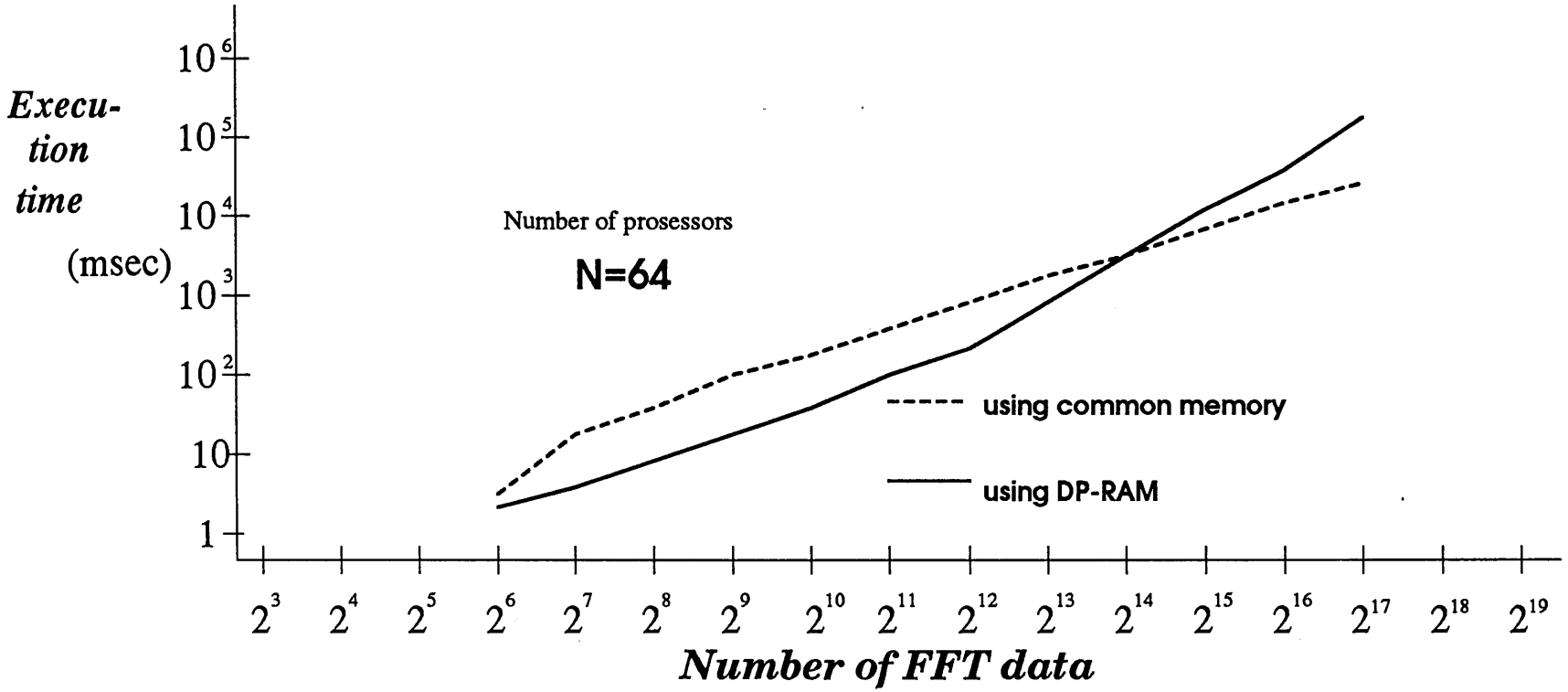



図１７ Comparison of execution types

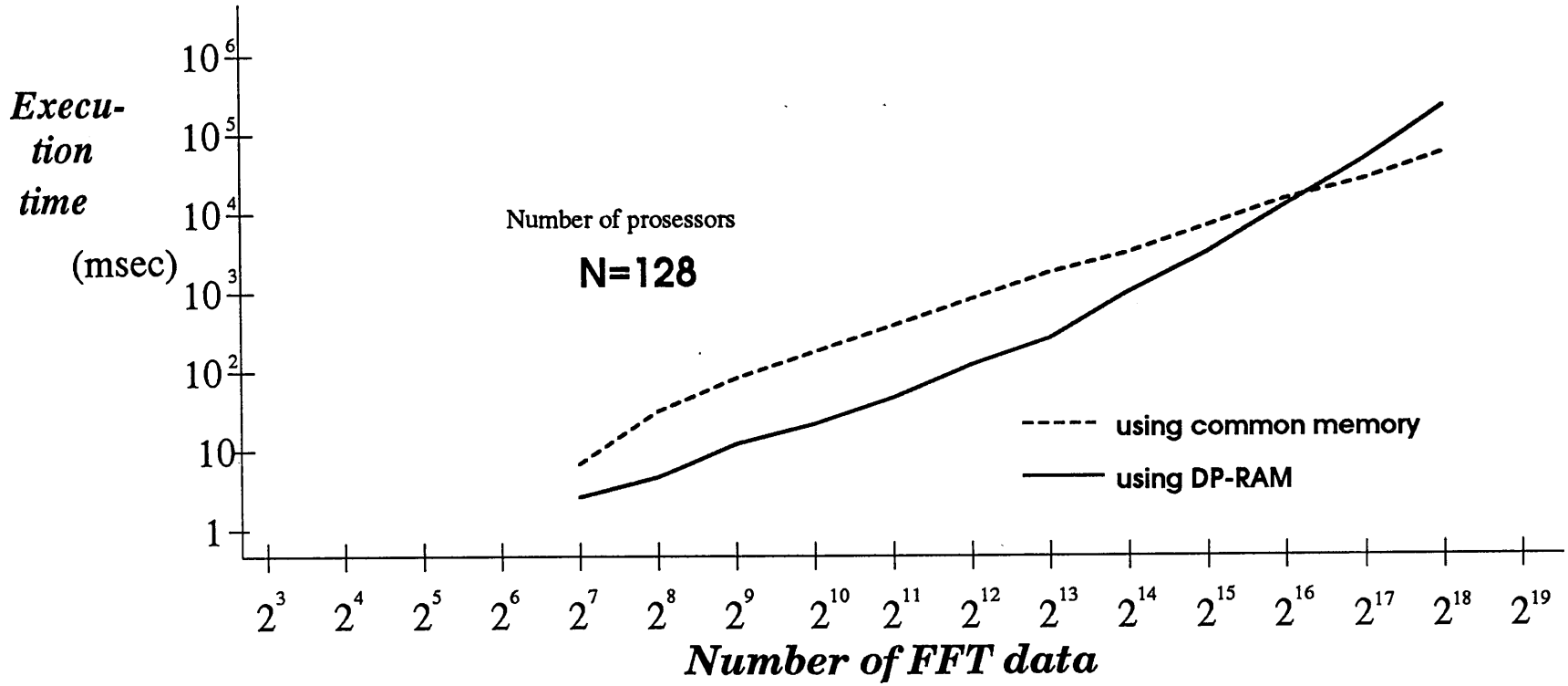



図１８ Comparison of execution types

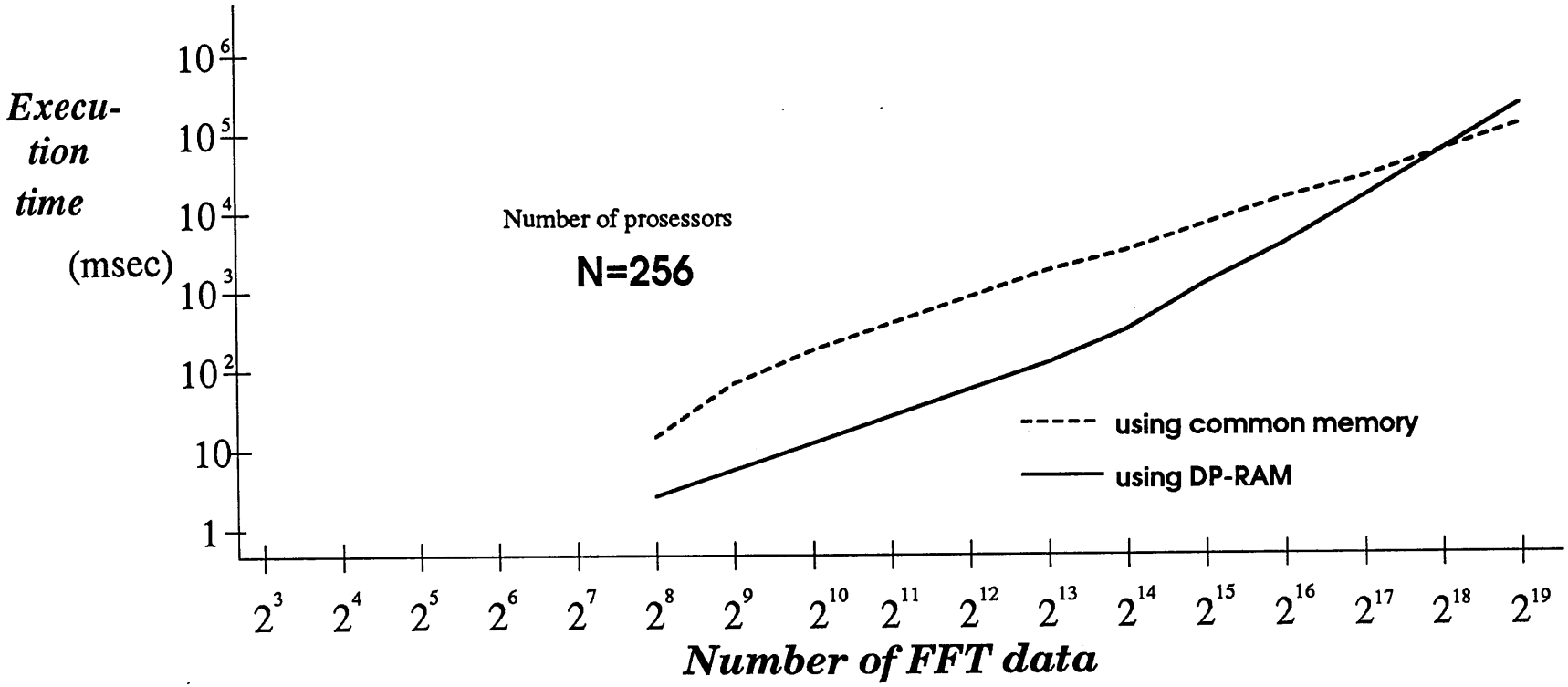



図１９ Comparison of execution types

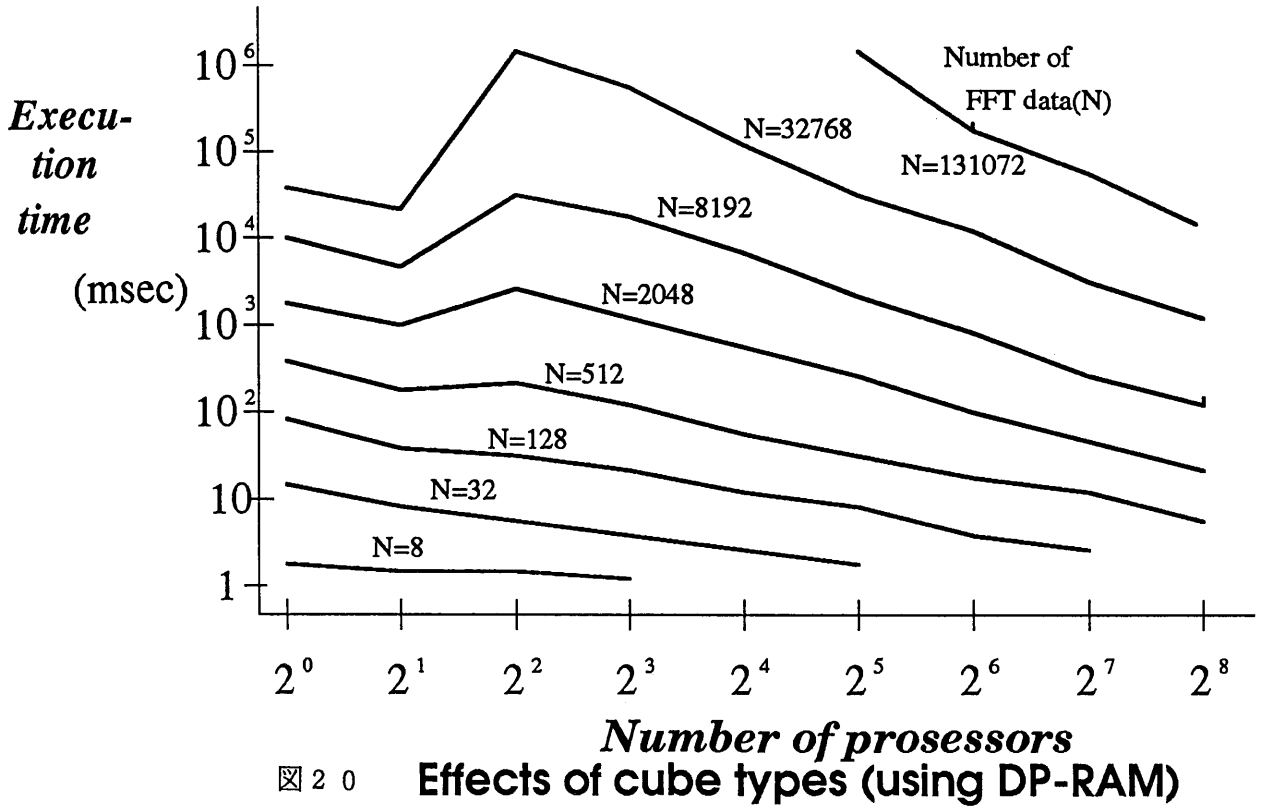



図２０ Effects of cube types (using DP-RAM)

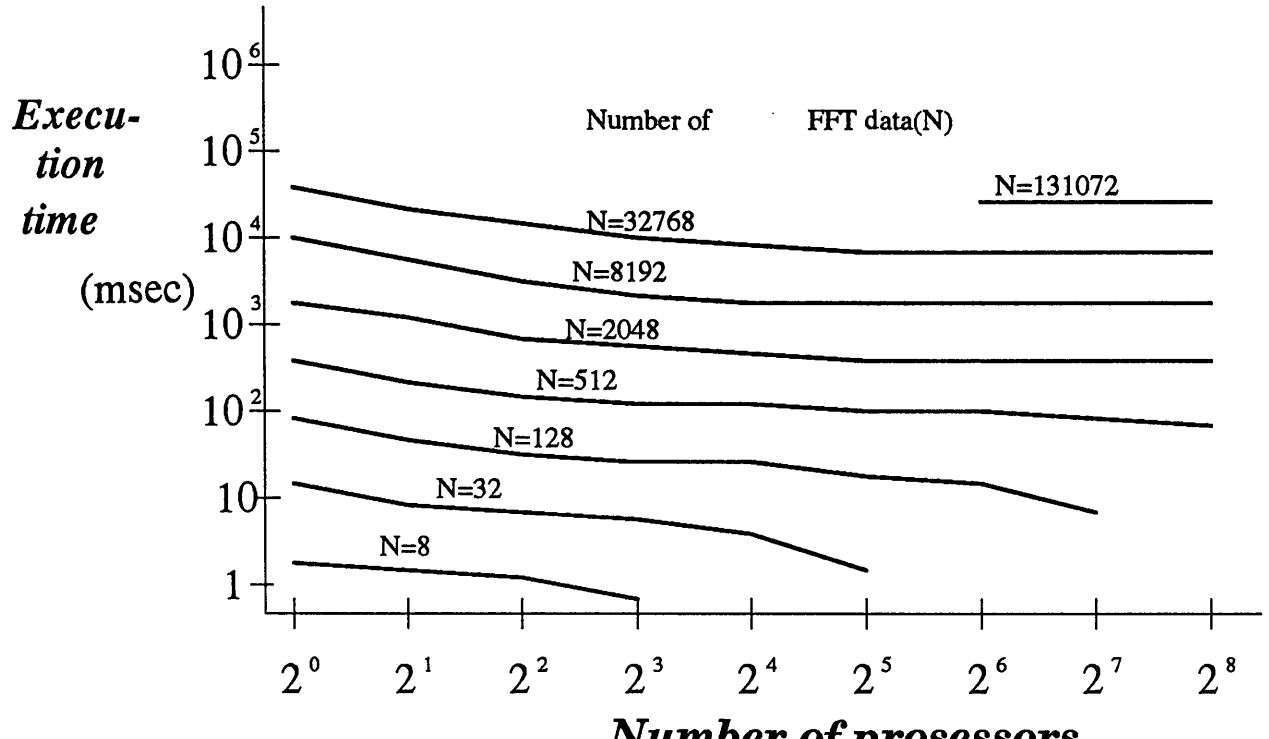



図２１ Effects of cube types (using common memory)

# 第5章　結論

本研究では、ハイパーキューブ型マルチプロセッサシステムにマルチポートメモリを取り入れることでデータ転送と処理を独立させ、そのシステムの性能評価を行うことを目的としている。

マルチポートメモリ型ハイパーキューブマルチプロセッサを採用することは、データ処理時間の短縮化にかなり有効であることが分かった。しかし、データ数がある値を越えてしまうとハードウェア的制限、すなわち、データ転送が512bytesのブロック単位でしかできないために、それ以上のデータを扱う場合にはデータ転送を複数回行わなければならず、そのことがボトルネックとなる。

一方、共有メモリを用いた場合には、プロセッサ数を増加させることによる速度向上はあまり期待できないことも分かった。

また、今回行ったシミュレーションでは、一回の演算で必要なデータ転送は隣接プロセッサ間の通信のみである。しかし、その他の問題においては、データ転送は任意のプロセッサ間で行われるために、その最短経路選択や故障ノード回避が行われれば、マルチポートメモリを採用したことによる効果は絶大なものになると考えられる。

# 謝辞

本研究を進めるにあたり、多くの御指導、御助言を戴いた阿曽弘具教授、下平博助手に心より感謝いたします。

また、堀口進助教授、井口寧氏には、本研究全般にわたり、多大な御指導、御意見、御協力を戴きました。ここに深く感謝いたします。

最後に、御討論、御協力を戴き、また日頃の生活においてお世話になった丸岡(東)研究室の皆様に感謝いたします。

# 参考文献

堀口　進：”マルチポートメモリ型・超高速ハイパーキューブマルチプロセッサシステムに関する研究”（July,1990）

梅尾博司：”並列計算機のためのアルゴリズムとアーキテクチャ”，bit, vol.23,No.6（1990）

富田真治：”並列計算機構成論”，昭晃堂

奥村晴彦：”C言語による最新アルゴリズム事典”，技術評論社