計算機アーキテクチャ 64－4
（1987. 1. 27）

並列推論マシンにおけるストリーム並列言語の実行方式の評価

垂井 俊明， 丸山 勉， 田中 英彦

（ 東京大学 工学部 ）

我々は、並列推論マシンでGHCを初めとするストリーム並列言語を実行する際に問題となる、共有変数へのアクセス等についての評価・検討を行なうために、ソフトウェアシミュレーションを行なった。シミュレーションでは、単一化プロセッサと共有メモリからなるユニットが多数台並列に接続されたモデルを仮定した。単一化プロセッサが他のユニットの共有メモリにアクセスする場合は、共有メモリネットワークを通じてアクセスが行なわれる。本稿では、そのシミュレーションモデルと処理方式について述べた後、シミュレーション結果について報告する。

A Preliminary Evaluation of the Execution Mechanism of Stream Parallel Languages

Toshiaki TARUI, Tsutomu MARUYAMA and Hidehiko TANAKA

Faculty of Engineering, University of Tokyo
3-1 Hongo 7-choume, Bunkyo-ku, Tokyo, 113 Japan

There are two important problems to execute stream parallel languages in parallel inference machine. One is the control of goals that are suspended and then activated. The other is the access frequency to shared variables. We made a simulation to evaluate these two problems. In this paper, first we describe the control mechanism of goals and management of shared variables, then we present the simulation results of these two problems.

1．はじめに

我々は高並列推論エンジンＰＩＥ[1] の開発を進めている。ＰＩＥはゴール書き換えモデルに基づき、論理型言語を高並列に実行する推論マシンである。現在迄のところ、ＰＩＥにおける並列実行は、OR並列を主体に検討が進められて来た。従って現在のＰＩＥのアーキテクチャそのままでは、ＧＨＣを始めとするストリーム並列言語を効率良く実行することは難しい。並列推論マシン上でストリーム並列言語を実行する際、問題になるのは次の２点である。

① 共有変数へのアクセスと負荷分散
② ゴール間の実行制御

ここではまず上記の①についてソフトウエアシミュレーションにより評価、検討を行なうことにする。具体的には、

・共有変数へのアクセスの頻度
・ＳＭ間ネットワークの構成
・負荷分散の方式

についての評価検討を行なう。

共有変数については、現在のＰＩＥで用いられている集中型の構造メモリ[2] を利用することも可能であるが、アクセスの集中等の問題が生じる可能性が大きいので、集中型に代わり分散型の共有メモリ（ＳＭ）を導入するのが適当であると考えられる。ＳＭと単一化プロセッサ（ＵＰ）が１つのユニットを構成し、各ＵＰが同一ユニット内のＳＭに高速にアクセスすることを可能にすることにより、他のユニットのＳＭへのアクセスをおさえ、高速な処理を行なうことができる。

シミュレーションの対象とする言語については、ゴール間の実行制御が不要であるＦＬＥＮＧを用いることにする。これにより①のみに対する評価をより明確にすることができる。

2．論理型言語ＦＬＥＮＧ

ＦＬＥＮＧ[3] はＧＨＣと良く似た言語であり、以下の点がＧＨＣとの主な相違点である。

① ガードが存在しない。従って定義節のヘッドリテラルの単一化が成功するとすぐにコミットされる。
② ボディ部の各リテラルは論理的 AND関係にない。従って、全てのボディ部のリテラルは必ず実行され、あるゴールが失敗しても他のゴールには影響を与えない。論理的な AND関係が必要な場合は、プログラム中に明示的に書く必要がある。

また、ＦＬＥＮＧにおいてもＧＨＣと同様に、ヘッドリテラルの単一化を行なう場合親ゴールの変数を具体化することはできず、この単一化はサスペンドする。

以上述べたように、ＦＬＥＮＧはＧＨＣと共有変数に対するアクセス等については同一であるが、実行制御の部分は大幅に簡略化されている。また、ＦＬＥＮＧはＧＨＣより低レベルな言語であるが、ほとんどのＧＨＣのプログラムはＦＬＥＮＧに容易にコンパイルすることができ、すでにコンパイラも完成している。

3．シミュレーションモデルの全体構成

図１に今回シミュレーションを行なうシステムの全体構成を示す。単一化等の処理を行なう単一化プロセッサ（ＵＰ）と共有メモリ（ＳＭ）の対が基本処理要素である推論ユニット（ＩＵ）を構成し、ＩＵが多数台並列に接続されてシステム全体を構成している。

システムにはＵＰ間を結び、基本処理単位であるゴールフレーム（ＧＦ）の分配等を行なう分配網（ＤＮ）、ＳＭ間を接続し共有変数の値や構造データ等のやり取りを行なったり、サスペンド状態にあるＧＦの起動を行なうための通信に用いられる構造メモリ網（ＳＭＮ）の２種類のネットワークがある。ＤＮについては、従来ＰＩＦで用いられてきた多段結合網[4] を用いる。各ＩＵ間での負荷分散は、ＤＮが各ＩＵの負荷に応じＧＦを自動的に分配することにより行なわれる。ＳＭＮについては現在のところどのような構成が最適であるか分からないため、とりあえず理想的な（接続時間一定、閉塞なし）ネットワークを仮定してシミュレーションを行なった。また同一ＩＵ内のＵＰとＳＭは密に結合されており、ＳＭＮを通さずに直接、高速にアクセスすることが可能である。これにより、負荷分散をうまく行なえ

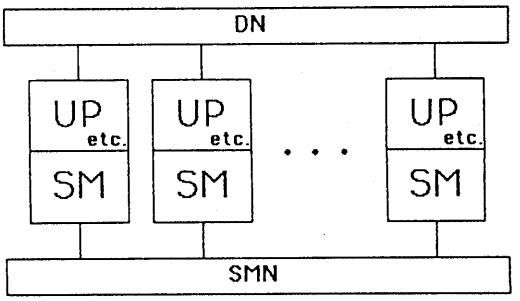

図 1 システムの全体構成

ばSMNへのアクセスの回数を減らすことが可能になる。

図2に推論ユニットの内部構成を示す。各部の機能は以下の通りである。

・DM（定義節メモリ）

プログラムを保持する。全てのDMは同じ定義節を持つ。

・UP（単一化プロセッサ）

単一化及びゴールの生成を行なう。UPからはDM、MM、Cache 、及び同一ＩＵ内のSMの内容を直接アクセスすることができる。他のＩＵのSMをアクセスする場合は、SMNを通してアクセスを行なう。

・MM（メモリモジュール）

ゴールフレーム（ＧＦ）の格納、管理等を行なう

・SM（共有メモリ）

共有変数及び構造データを記憶する。

・Cache(キャッシュ)

他のＩＵのSMの内容のキャッシュである。ここで言うキャッシュとは、外部SMのアドレス（ＩＵ番号、ＩＵ内アドレス）によりSMの内容を検索することのできる連想メモリである。Cache 中には他のSMの中の共有変数の値、構造データ、及び undefであるセルのアドレスが記憶される。

4．シミュレーションにおける処理方式

4．1　ゴールフレームの表現

今回の処理モデルでは、処理の基本単位となるゴールフレーム（ＧＦ）はリテラル毎に分割されている。従って単一化の結果、新たに（ AND関係にある）複数のリテラルが生成された場合、それらは別々のＧＦとして異なるMMに分配される。その際SMNへのアクセスを減らすため、ＧＦを生成するUPと同じ（ローカルな）SM中にある構造データは、ＧＦの構造部としてＧＦのリテラル部と共にコピーされる。また、同様にローカルなSMへのポインタについても、デリファレンスした結果がＧＦ中にコピーされる。

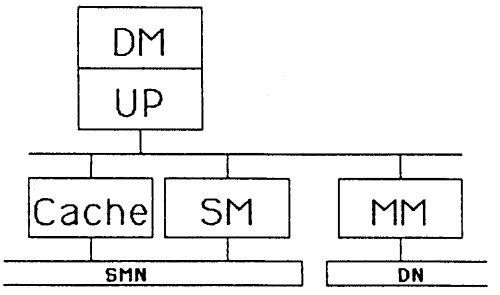

図 2　ＩＵの内部構成

4．2　共有変数の配置と単一化

ＵＰにおいてＧＦの処理を行なうとき、定義節中の変数は全てローカルなSM中におかれ、 undefに初期化される。そしてヘッドリテラルの単一化中におきた結合は、その変数セルに書き込まれる。定義中の構造データが結合された場合は、構造データはSM中にコピーされ、変数セルにはそこへのポインタが書き込まれる。新ＧＦの生成時には、変数セルのうち値がバインドされているものについては、その値がＧＦ中にコピーされ、 undefのままのものについては、そのセルへのポインタがＧＦ中に書き込まれる。

4．3　単一化のサスペンドとアクティベイト

ＦＬＥＮＧにおいて単一化がサスペンドするのは以下の２つの場合である。

- ヘッドリテラルの単一化中に、親ゴールの undefの変数を具体化しようとしたとき
- 定義節のヘッドの bind-to-non-variable annotationのある変数に undefの変数を単一化しようとしたとき

サスペンドしたＧＦは undefの変数に値がバインドされると再び実行することができるようになる。ＧＦのサスペンド及びアクティベイトの処理は以下のように行なわれる。

①ヘッドリテラルの単一化中に、サスペンドが起こるようなバインドをしようとすると（実際にはバインドをしようと undefのセルを読みに行った時点で）、MM中のＧＦのアドレス（ＩＵ番号、ＩＵ内アドレス）がSM中のサスペンドの原因となった undefのセルの所に記憶される。

②ＧＦの単一化において、１つの単一化も成功せず、１つ以上の単一化がサスペンドした場合、ＧＦはMM中のアクティブなキューから外され、サスペンド状態になる。

③その後 undefのセルに値がバインドされると、ＳＭはサスペンドしていたＧＦにSMNを通してアクティベイトコマンドを送り、ＧＦは再び実行される（それと同時に新しい値が Cacheに書き込まれる）。

ＵＰで実行中のＧＦに対してアクティベイトコマンドが来た場合は、ＵＰは実行を中止し、 Cache中の新しい値を使ってＧＦの処理をやりなおす。

4．4　Cache における処理

各ＩＵは他のＩＵのＳＭの内容のキャッシュを持つ。ＵＰが他のＳＭのデータをアクセスする場合は、先ずCache の内容を調べ、該当するエントリがある場合（ヒットした場合）はその値を用いて処理が続けられ、外部ＳＭへのアクセスは行なわれない。Ｃacheがヒットしなかった場合にはＳＭＮを通じて値の読み出しが行なわれ、読み出された値は Cacheに登録される。また、他のＳＭ中の undefのセルのアドレスについても、１つのＧＦの処理が終了するまでここに記憶される。

4．5　システム述語 unifyにおける処理

ＦＬＥＮＧにおいて親ゴールの変数への値の書き込みは、システム述語 unifyにより行なわれる。その際に、もしＵＰが書き込もうとしたＳＭ中の変数セルに既に他のＵＰにより値が代入されていた場合は（書こうとしたセルが undef以外だった場合）、ＵＰは書き込まれていた値を読み出し、その値と書き込もうとした値との単一化を行なう。

4．6　負荷分散の方式

ＩＵ間の負荷分散はＤＮが各ＩＵの負荷（ＩＵ内のＧＦの数）に応じ、ＧＦを分配することにより行なわれる。ＵＰにより生成されたＧＦは、接続可能なＩＵのうちで一番アクティブなＧＦの数が少ないＩＵ（アクティブなＧＦの数が同じ時は、一番サスペンドしたＧＦの数が少ないＩＵ）に送られる。ただし、同一ＩＵ内のMMにアクティブなＧＦがない場合は、ＧＦはＤＮを通さずにローカルなMMに戻される。

5．シミュレータでの仮定

シミュレーションに当っては以下のことを仮定した。

① ＵＰは現在のＰＩＥの試作ＵＰとほぼ同程度の機能のハードウェアを仮定し、マイクロ命令レベルまでシミュレートする。

② ネットワークは一定時間でかならずつなげる閉塞のない理想的な回線交換網とする。

③ ＳＭＮをつなげるのに要する時間については、０～20（クロック）の間で変化させ、接続時間の実行時間への影響を調べる（それ以外の測定では 5クロックとした時のデータを測定した）。

④ Cache は連想メモリであり、必要なデータを高速に検索することができる。

⑤ Cache の容量は十分大きい。

⑥ ＳＭ、MMはマルチバンクになっており、外部からのネットワークとローカルなＵＰから同時に読み出しを行なうことができる。ただし、書き込みを行なう場合にはロックがかけられる。

シミュレータはUNIX上のＣ言語で実装し、約10000行である。

シミュレーションに使用した例題は以下の４っつである。

- nreverse120　(NREV)－長さ 120のリストの反転
- qsort64　(QSORT)－長さ64のリストのソート
- primes600　(PRIMES)－ 600までの素数を求める
- perm5　(PERM)－長さ 5のリストの順番を入れ換えた結果を全て求める（全解探索）

6，シミュレーション結果

6．1　並列度

ＩＵを 1～64台まで変えた場合の台数効果を図 3に示す。また、ＩＵ64台の時の各例題の稼動ＩＵ台数、及びＧＦ数の変化を図 4～図11に示す。また、表 1にサスペンド、アクティベイトの回数を示す。

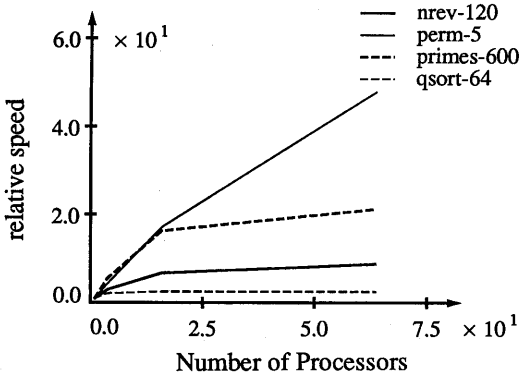

図 3　台数効果

	nrev120	qsort64	primes600	perm5
num. of suspended goals	539	584	20246	0
num. of hook	701	1644	41811	301
num. of activate command	579	1378	34701	301
suspended in MM	539	584	20246	0
executed by UP	31	62	1285	23
no goals to activate	9	732	13170	278

表 1　サスペンド、アクティベイトの数

PERMはOR並列な全解探索問題なので、サスペンド等は生じておらず、ストリーム並列処理は行なわれていないため、非常に高い並列度が得られている。PRIMESは素数のフィルタがストリーム並列に動くため、比較的大きい並列性を持っており、suspendするGFも多い。NREVは前半のリストを分割する部分には全く並列性は見られないが、後半のappendによりリストを結合する部分は、多数の unifyが並列に動くため高い並列度を示す。またゴール数のグラフ（図 8）からは、前半のリストの分割が終った所で、appendが 120個サスペンドしていることがわかる。 QSORTも同様に最後のリストを連結する部分の並列度が高くなっているが、差分リストを使っているので並列度はそれほど高くならない。

6．2　SMへのアクセス数

表 2にSMへのアクセス数を示す。NREVやPRIMES等では定義節の変数以外へのローカルSMへのアクセスが外部SMへのアクセスの倍程度あり、共有変数へのアクセスを考慮に入れた負荷分散を行なっていないにもかかわらず、ある程度のローカリティがあることがわかる。PERMで外部SMへのアクセスが非常に少ないのは、PERMはもともとORパラレルな問題であり、解の収集の時以外には外部SMへのアクセスがほとんど行なわれないからである。

	nrev120	qsort64	primes600	perm5
total access to SM	131499	22086	337887	91465
access to external SM	29397	2647	84575	978
access to local SM	102102	19439	253312	90487
access to Def's variable	29161	4267	101346	13535

表 2　SMへのアクセス回数

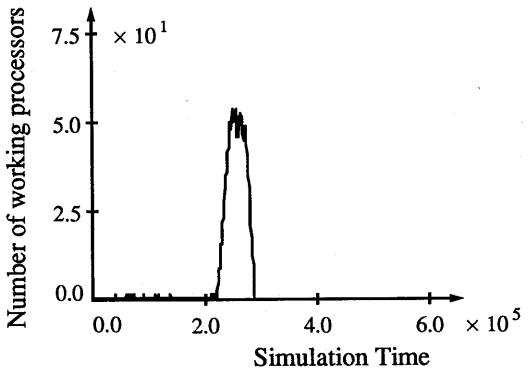

図 4　稼動プロセッサ台数の変化(NREV)

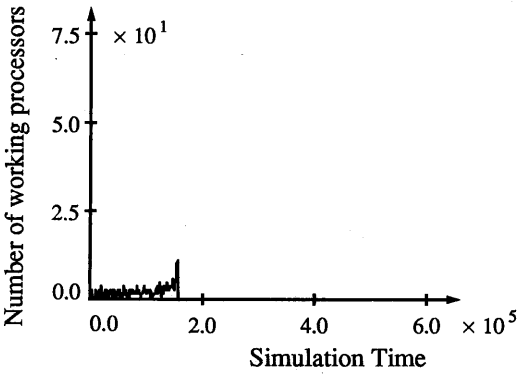

図 5　稼動プロセッサ台数の変化(QSORT)

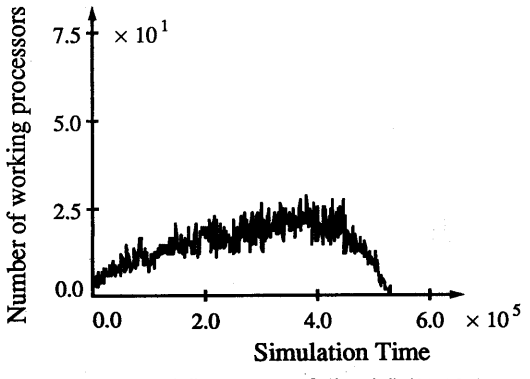

図 6　稼動プロセッサ台数の変化(PRIMES)

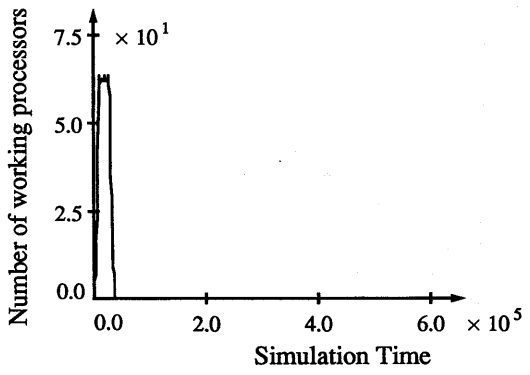

図 7　稼動プロセッサ台数の変化(PERM)

6．3　キャッシュの効果

表 3に外部ＳＭへの読みだしと Cacheのヒット数を示す。 Cacheのヒット数は多くて 2～ 3割であり、そのうち undefのヒットが半分程度を占めている。また注目すべき点として、以前に実行した別のＧＦがエントリしたデータがヒットすることはほとんどないことがわかる。これは、外部ＳＭへのポインタは、一度値が読出されるとその値は新ＧＦの生成時にＧＦ中にコピーされるため、ほとんどのポインタは１回程度しか読まれないためである。従って Cacheがヒットするうちのほとんどは、ある入力ゴールに対して複数の定義節の単一化を行なっているときに生じるものである。従って、 undef以外の Cache内のデータも１つのＧＦの実行が終った時点で消去してしまって良いと考えられる。

	nrev120	qsort64	primes600	perm5
external SM read	22143	1989	62866	498
total cache hit	907	737	19700	18
undef	436	320	6430	0
except undef	471	417	13270	18
entried by other goals	8	2	44	0

表 3　キャッシュの効果

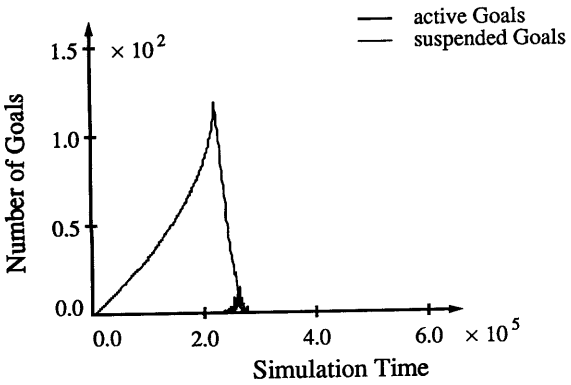

図 8　ＧＦの数の変化(NREV)

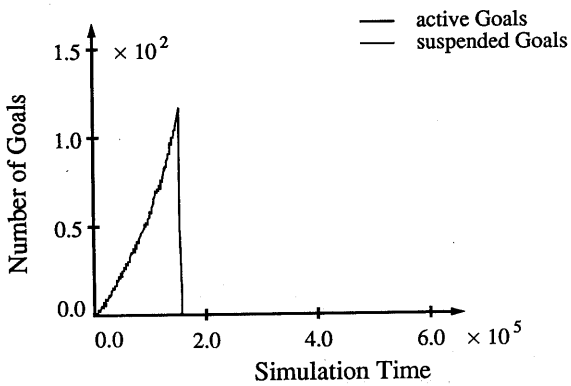

図 9　ＧＦの数の変化(QSORT)

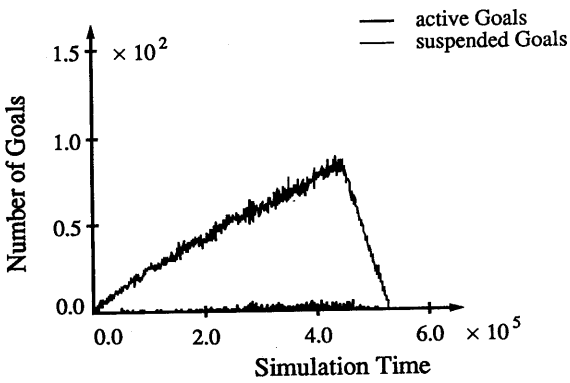

図10　ＧＦの数の変化(PRIMES)

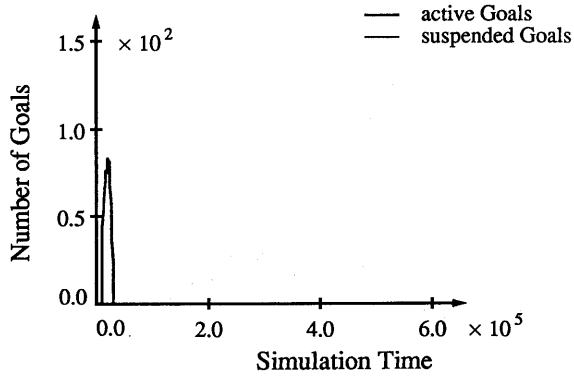

図11　ＧＦの数の変化(PERM)

6．4　SMNの接続時間の影響

図12にSMNをつなげるのに要する時間を 0～20（クロック）と変化させた場合の実行時間の変化を示す。PERMや QSORTのような外部SMへのアクセスが比較的少ない例題では実行時間はほとんど変化しないが、PRIMESのように外部SMへのアクセスやサスペンド等が多い例題では速度にかなりの変化が見られる。従ってSMNの接続時間はストリーム並列計算機の性能にかなり影響を与えることになる。

6．5　SMNへのアクセスの頻度

ここではSMNへのアクセスが多いNREVとPRIMESの結果により検討を行なう。図13、図14にSMNへの同時アクセス数の最大値と平均値の時間変化を、図15～図18にSMNへの読み出し要求と書き込み要求の頻度を示す。NREVの後半のように、64台ほとんど全てのプロセッサが単一化を行なっている場合には、SMNを通じた外部SMへの読み出し要求が約 2クロックに１回、書き込み要求が約 6クロックに１回生じている（これは、１回の単一化に約50クロックかかっていることからも理解できる値である）。そしてその時にはSMNに最大12個（平均でも 6個）の同時アクセスがおきている。（１回当りの平均アクセス時間は、SMNを接続する時間が 5クロックの時で約10クロックである。）従ってこの方式で64台のＩＵをつなげるとSMNへのトラヒックはかなり多くなることがわかる。

7．おわりに

以上のように、シミュレーション結果よりSMNへのトラヒックはかなり大きく、SMNの構成によってシステム全体の性能が左右されることがわかった。SMNの構成についてはバス、多段網等が考えられるが、シミュレーション結果より、SMNをバスで接続した場合は接続可能なＩＵの台数はせいぜい16台程度であり、システムを階層構造にする必要がある。また、SMNを多段網にした場合の性能についても、より詳細に検討してみる必要がある。今後は

- SMNの構成の検討
- 共有変数の他のアクセス方法（マルチリード・ワンライト方式等）についての検討
- より効率的な負荷分散方式についての検討。

等を行なって行きたい

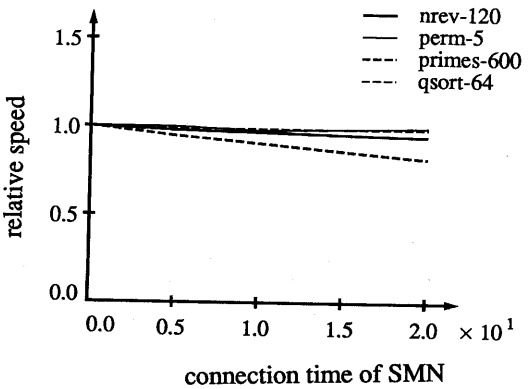

図12　SMNの接続時間の影響

＜参考文献＞

[1] Moto-oka, T, Tanaka, H, et al , "The Architecture of a Parallel Inference Engine －ＰＩＥ－", FGCS '84, ＩCOT.

[2] 平田、田中、元岡：”ＰＩＥにおける構造メモリの構成について”、情報処理学会、アーキテクチャワークショップインジャパン '84、1984.

[3] Nilsson, M, Tanaka, H, "－FLENG Prorog－ The Language which turns Super-computers into Parallel Prolog Machines", The Logic Programming Conference '86, ＩCOT.

[4] 坂井、田中、元岡：”動的負荷分散を行なう互結合網”、信学技報、1985年 8月.

[5] 垂井、丸山、田中：”ＰＩＥにおける並列論理型言語ＦＬＥＮＧの実行方式”、情報処理学会第33回全国大会、5B-4、1986.

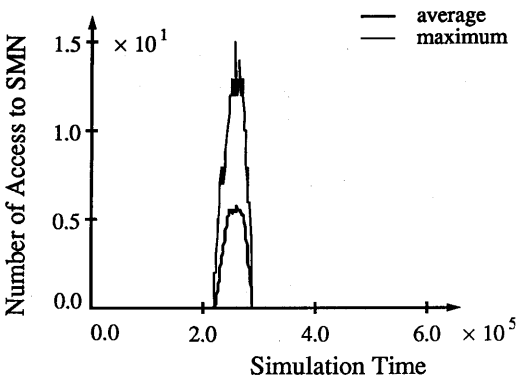

図13　SMNへの同時アクセス数(NREV)

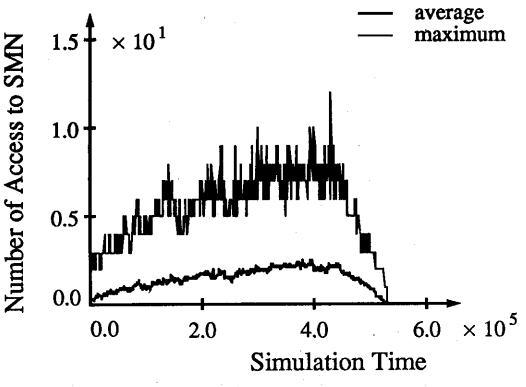

図14　SMNへの同時アクセス数(PRIMES)

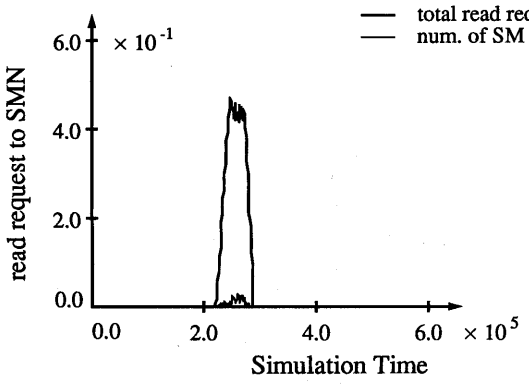

図15　SMNへの読み出し要求(NREV)

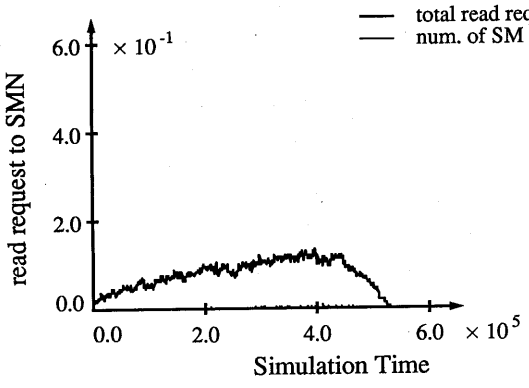

図16　SMNへの読み出し要求(PRIMES)

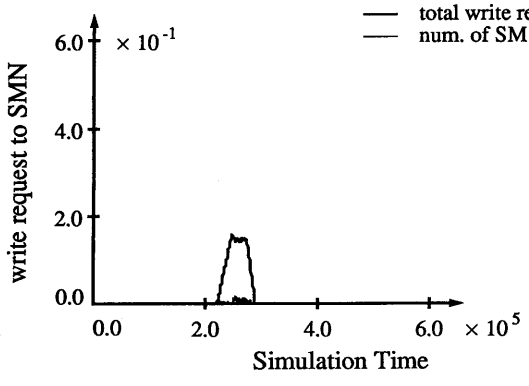

図17　SMNへの書き込み要求(NREV)

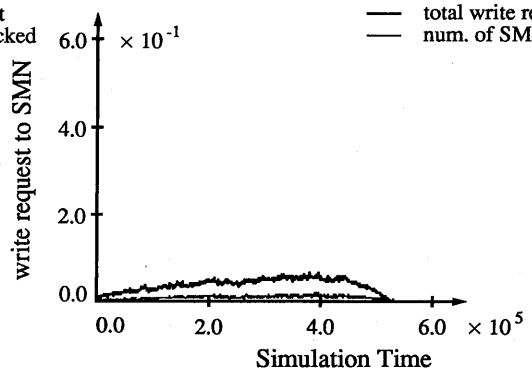

図18　SMNへの書き込み要求(PRIMES)