計算機アーキテクチャ 89−27
（1991. 7. 19）

シミュレータによるVPIM処理系の評価
— データの輸出入処理に関して —

高木常好
新世代コンピュータ技術開発機構
仲瀬明彦
東芝 総合研究所

我々は並列推論マシンPIMのための論理型言語KL1の分散処理系VPIMを開発している。VPIMにおけるクラスタ間処理は、「GCはできる限り即時に行なうこと」、「メッセージ通信の頻度を抑えること」を基本方針として設計し、シミュレータを用いてその評価を行なっている。今回の評価では、(1)インクリメンタルな回収を行なうことにより、輸出入表のために必要な領域を1／10にできた、(2)ユニフィケーションを最適化することにより、ユニフィケーションのためのメッセージを最大で50％削減できた、という結果を得た。

Evaluation of VPIM: A Distributed KL1 Implementation
— Focusing on Inter-cluster Operations —

Tsuneyoshi TAKAGI
Institute for New Generation Computer Technology (ICOT)
Mita Kokusai Bldg. 21F, 1-4-28, Mita, Minato-ku, Tokyo 108 Japan.
Akihiko NAKASE
TOSHIBA R&D Center

We are developing a distributed KL1 inplementation on parallel inference machines(PIMs). The design policy on inter-cluster operations are "Garbage data should be reclaimed immediately", "Frequency of issuing inter-cluster messages should be kept low". Now, we are also evaluating its validity on a simulator. This time, our evaluation results show as follows: (1) By the incremental reclaimation, the size of export/import-table areas were reduced to 1/10. (2) By the optimization of unify operation, at most 50 percents of unification messages could be eliminated.

1 はじめに

ICOTで開発中の並列推論マシンPIM[1]は、クラスタをネットワークによって疎に結合した構成になっている。クラスタは、共有バス／共有メモリによって密に結合された10台程度の要素プロセッサ(PE)からなる。図1にPIMの構成を示す。PIMのような大規模システムでは、システム全体が同期して行なう大域ガーベジ・コレクション(GC)を起こすことは望ましくない。そこでクラスタ毎の一括GCを可能にするため、クラスタ間のデータ参照では内部アドレスに依存しないようなアドレス変換機構を用いる。また、クラスタ間のデータの移動ではメッセージを用いたクラスタ間通信を行なう。

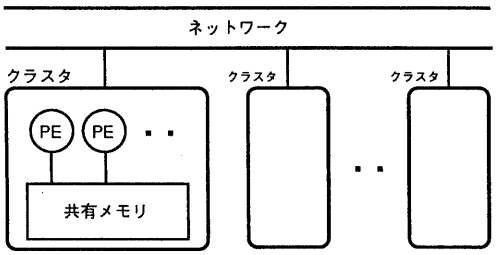

図 1: PIMの構成

PIMのプログラミング言語であるKL1[2]は、高機能な並列論理型言語である。処理系を作る上では言語と機械のマシン命令との間にある大きなギャップを埋めるために、中間言語として抽象機械語KL1-B[3]を導入した。我々はこのKL1-B言語処理系を、仮想マシン上で開発している。この処理系をVPIM(Virtual PIM)[4]と呼ぶ。
クラスタ間処理部の設計においては、

- GCはできる限り即時に行なう。
- メッセージの頻度を抑える。

という目標を設定し、様々な工夫を導入している。また、我々はVPIMをC言語に変換してシミュレータを構成し、それを使用して各種データを収集し処理系の評価を行なっている。

本稿ではVPIMのクラスタ間処理について評価した結果を報告する。2章ではVPIMにおけるクラスタ間処理について述べ、3章ではテストを行なった状況を説明し、シミュレータによって得られたデータからクラスタ間処理の各部について評価する。

2 VPIMのクラスタ間処理

2.1 設計における基本方針

(1) GCはできる限り即時に行なう

KL1のような論理型言語は破壊的代入ができない。したがって、その処理系の設計においてはメモリの消費速度を極力抑えることが重要な課題の一つである。しかし、いづれはメモリを消費してしまうのでGCを起こさなくてはならないわけであるが、システム全体が同期して行なう大域GCはオーバーヘッドが大きいので避けなければならない。そこでクラスタ間に渡るデータの参照を、クラスタの内部アドレスには依存しないアドレス変換機構を用いて、クラスタ毎に独立したGCができるようにする。またクラスタ間に渡るデータの参照がなくなった時点で、アドレス変換機構で使用したセルを即時に回収する。

(2) メッセージ通信の頻度を抑える

一般的にネットワークを介したメッセージ通信にかかるコストは高い。またネットワーク・トラフィックの混雑も避けなければならない。したがって、様々な最適化を施して、できる限りメッセージ通信の頻度を抑える。

2.2 ゴールの送出とデータの外部参照

PIMでは小粒度プロセスである**ゴール**が実行単位であり、そのゴールを他のクラスタに送出することによってクラスタ間の負荷分散を行なう。クラスタ間ではメモリを共有していないので、ゴールが送出されて他のクラスタで実行される時は実際にデータが移動することになる。またKL1は破壊的書き込みを許さないので、ゴールの引数が既定義データの場合はコピーを作ることができるが、具体化されていない変数の場合は変数の同一性を保つためにその変数へのポインタを生成し、それをゴールの引数として送出する。このようなクラスタ間にまたがるデータの参照を**外部参照**と呼び、ゴール送出時に生成した変数へのポインタを**外部参照ポインタ**と呼ぶ。また、外部参照ポインタによって指されるデータを**外部参照データ**と呼ぶ。そして外部参照ポインタを生成し、他のクラスタへ送出することを**データの輸出**と呼び、外部参照ポインタを他のクラスタが受け取ることを**データの輸入**と呼んでいる。

ゴールは`throw_goal`メッセージによって他のクラスタに送出される。なお、ゴールの引数が既定義であってもそれが構造体である場合には、外部参照ポインタを送出することにしている。行き先クラスタでその引数の値が用いられるとは限らないし、データのコピーにかかるコストが無視できないからである。

データを輸入したクラスタで外部参照ポインタが指すデータの値が必要になった場合には、`read`メッセージが輸出元に送られ、輸出元でその変数の値が定義された時点で`answer_value`メッセージにより値の返信(値のコピー)が行なわれる。また、輸入した外部参照ポインタへのアクティブ・ユニフィケーションでは`unify`メッセージが送信され、輸出元でユニフィケーションが行なわれる。表1に代表的なクラスタ間メッセージを示す。

表 1: 代表的なクラスタ間メッセージ

メッセージ	機能
throw_goal	ゴールの送出
read	外部参照データの読み出し要求
answer_value	read メッセージに対する返答
unify	外部参照データのユニファイ要求
release	外部参照ポインタの WEC 返却

2.3 輸出入表

クラスタ毎の一括 GC では、輸出した外部参照ポインタもマーキング・ルートにしなければならないので、これを管理するため輸出表を設けている。また、2.4で述べる一括 GC 時に他クラスタの輸出表エントリを回収するために輸入表を設けている。

図 2に輸出入表を利用した外部参照方式を示す。一括 GC ではクラスタ内部でのデータの移動が他のクラスタに影響を与えないようにするため、外部参照ポインタをクラスタの内部アドレスとは独立な < n, e > (n は輸出クラスタ番号、e は輸出表におけるエントリ位置) という形で表現する。またクラスタ内はアドレス空間を共有しているので、輸出入表はクラスタ毎に置いている。

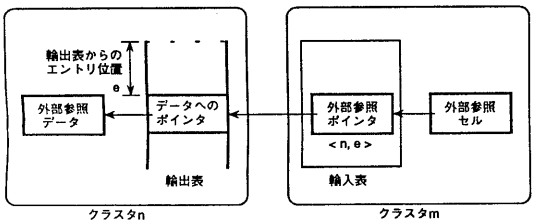

図 2: 外部参照方式

輸出入処理は、データを輸出する度に輸出表エントリを割り付けて行なえば、処理コストも低く簡単である。しかし、同一データを何回も輸出する場合には、輸出する度に別の輸出表エントリを割り付けてしまうので、輸入側では全て異なったデータに見えてしまう。このため、同一データを n 回輸出したら n 回の read メッセージが送出される恐れがある。これを防ぐために、重み付きリファレンス・カウンティング方式を応用した、**WEC**(Weighted Export Counting) 方式[5] を用いて多重参照データ用の輸出入表を別に設けている。

図 3に単一参照データを輸出した状態を示す。輸出表にはデータへのポインタを、輸入表には外部参照ポインタを登録する。

図 4は、クラスタ n 内において多重参照であったデータを、クラスタ m に対して 3 回輸出した状態を示している。輸出入表は WEC などを保持するため複数ワードのセルを使用する。輸出表にはデータへのポインタと WEC を、輸入表には外部参照ポインタ、WEC、外部参照セルへのポインタを登録する。

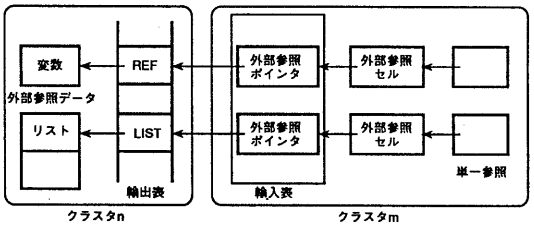

図 3: 単一参照データの輸出

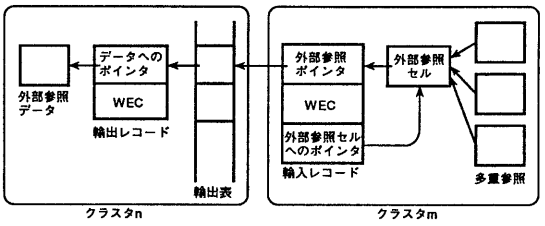

図 4: 多重参照データの輸出

2.4 WEC を用いた即時 GC と一括 GC

ある輸出表エントリが輸入側のクラスタで参照されなくなったことを検出し、それを即時に回収することによって大域 GC の頻度を削減することができる。このクラスタ間の即時 GC を行なうのが WEC 方式である。WEC はデータが輸出表に登録された時点に与えられる整数値で、ある外部参照ポインタに対して、それを持つ輸出入表エントリとネットワークメッセージ中の値の合計が 0 になるように分配される。輸出表エントリは負の数を、その他は正の数を持っている。

外部参照セルがデータのコピーによって具体化され、輸入側で使用済みとなった外部参照ポインタの WEC は release メッセージによって輸出元に返却される。そして、輸出元のエントリが持つ WEC の合計が 0 になった時、このデータに対する参照がなくなったことを意味し、輸出表エントリが回収できる。この方式は通常のリファレンス・カウント方式に比べてカウント値のメンテナンスのためのメッセージ通信量を半分程度にできるという特徴がある。

一括 GC 時においても、マーキング後に輸入表をたどって参照されていない外部参照セルを見つけると、やはり輸入表エントリを回収して、輸出元に release メッセージを送出する。release メッセージを受信した輸出元は WEC の計算を行ない、その値が 0 になった時に輸出表エントリを回収する。

2.5 分割輸出

分割輸出とは、外部参照ポインタを別のクラスタに輸出するときに生じる。所持していたWECの半分が新しい外部参照ポインタが付けられる。図5に分割輸出の様子を表す。もし分割輸出をしなかった場合、クラスタmとクラスタkの間に輸出入表を作り、クラスタkからはデータの実体を直接参照できなくなってしまう。WECの効果はこの分割輸出が多く出た時に大きい。なぜなら、もし通常の参照カウンティング方式を用いたとすれば、分割輸出の度に輸出側へカウンタのメンテナンスをするためのメッセージが送信されることになるが、WEC方式では輸出表を操作することなく外部参照ポインタが輸出できる。

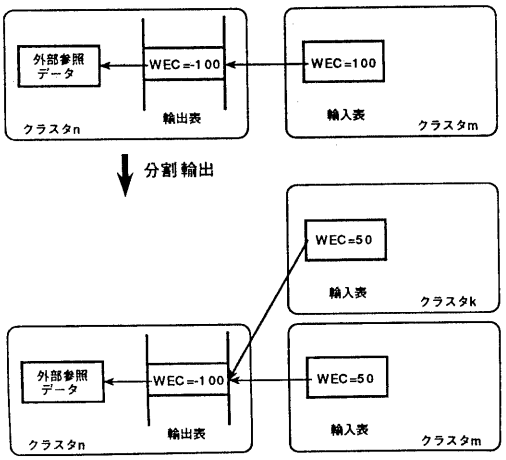

図 5: 分割輸出

2.6 変数セルと外部参照セルのユニフィケーション

変数セルと外部参照セルをユニフィケーションすることは、外部参照セルを指すポインタを変数セルに書き込むことである。しかし図6に示すような構造の場合においては、双方のクラスタでポインタの書き込みを行なうとクラスタ間に渡るループ構造ができてしまい、無限の参照を繰り返すことになる。そこで変数セルにポインタの書き込みを行なって良いかどうかを示す属性を外部参照セルに持たせている[6]。属性にはSafeとUnsafeがあり、Safe属性ならばポインタの書き込みができ、Unsafeならば外部参照セルの輸出元に対してunifyメッセージを送出し、ユニフィケーションを依頼する。

また、unifyメッセージ送出の頻度を下げるために以下にあげる最適化をしている。

(1) 輸出済み変数セル

上述のように変数セルが他のクラスタから参照されている場合は、ループ構造を作らないようにSafe/Unsafe属性を調べる必要がある。しかし、Unsafe属性であっても変数セルが他のクラスタから参照されていないことが明らかであれば、Safe/Unsafe属性に関係なくループ構造を作ることがないので、ポインタの書き込みによるユニフィケーションが行なえる。そこで、全ての変数セルに他のクラスタからの参照の有無を区別するためのタイプを設けている。

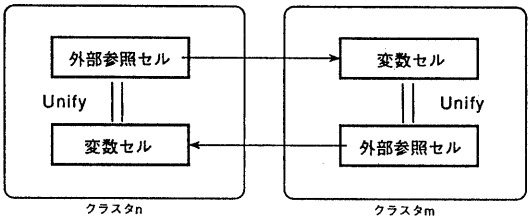

図 6: クラスタ間のループ構造

(2) 構造体データの外部参照

構造体データはデータのコピーにかかるコストや消費されるメモリが無視できないため、その値が必要になった時はじめてコピーをする。したがって、ゴールの送出時には構造体へのポインタのみを送信する。また、図6のようなループ構造ができるのは外部参照ポインタの先が変数セルの時だけなので、構造体へのポインタを外部参照ポインタとして送出している場合は起こらない。このため、輸出するポインタが変数でないことを知らせるために、外部参照ポインタのタイプに区別をつけている。

(1)、(2)の最適化を入れたことによって、ポインタの書き込みだけではユニフィケーションができず、unifyメッセージを送出しなければならないのは、Unsafe属性の外部参照セルと輸出済み変数セルの組合せだけになる。

3 計測・評価

3.1 シミュレータの構成

VPIMをC言語に変換し、PIMのハードウェアの構成要素をシミュレートする部分は直接C言語で記述した。そして、これらをコンパイル、リンクすることによってシミュレータを構成する。

PIMは、複数のPE(Processing Element)が共有メモリを介してつながるクラスタ構造と、複数のクラスタがネットワークにより結合された構造との2階層構造を持つため、シミュレータにおいてもこの構造を実現する。また、VPIMをマシン命令の変わりにC言語に変換したとしても、その周辺を構成するハードウェア群の部分をシミュレータ上に実現する必要がある。

3.2 ベンチマークプログラム

評価に使用したKL1プログラムは以下にあげる4プログラムである。

- **Pentomino:** 詰め込みパズルの解をすべて求めるプログラムで、長方形の枠の中に様々な形の部品をピッタ

リはめ込む方法をすべて探索する。このプログラムでは部品の置き方に関するOR木の全解探索を行なう。0番クラスタがルートノード、即ち空の状態から始めて探索木のある深さまで進み、そこから先のサブ木を各クラスタにできるだけ公平に分配するという動的負荷分散を行なう。

- **Bestpath:** グリッド状のネットワークでの最短経路を求める。ネットワークの各ノード間は非負のランダムなコストを持つエッジで結ばれており、あるノードを出発点としたそれ以外への全てのノードに対する最短経路を求める。このプログラムではまず乱数を用いて問題とするグリッドを生成し、その後最短経路を求める処理が実行される。負荷分散は静的に行なっている。

- **Pax:** 自然言語の構文解析プログラム。このプログラムでは左隅構文解析方に基づいて、解析木の各レイヤーに現れる各ノードを並列に実行する。

- **Tsumego:** 囲碁で、ある局面を与えた時に勝つための次の一手を求める。このプログラムではまず全ての状況をある深さで探索し、その後は各状況を負荷分散して、アルファ・ベータ探索手法を用いた最適解を求める。

計測ではこれらのプログラムをクラスタ内PE台数を1台で、それぞれクラスタ台数を2、4、6台にして行なった。表2に各プログラムの総リダクション数を示す。クラスタ数が増すと総リダクション数も増加しているのは、ゴールの送受信やユニフィケーションの再試行の簡素化のためにリダクション機構を利用しているからである。ただし、Tsumegoでリダクション数が増えていないのは非決定的なプログラムであるからである。

表2: 各プログラムにおける総リダクション数

	Pentomino	Bestpath	Pax	Tsumego
2 CL	189K	141K	146K	58K
4 CL	191K	145K	146K	52K
6 CL	191K	150K	147K	53K

3.3 評価項目

2章で述べた、VPIMの処理方式が有効に機能するかどうかを見るために、以下にあげる項目の計測を行なった。

(1) 輸出入表エントリのインクリメンタルな回収

輸入表エントリは外部参照セルが具体化された時点で回収できる。また、輸出表エントリは参照カウントが無くなった時点で回収できる。これらの回収がどのくらい起きているかを調べるために輸出入表の使用状況を、割付けられたエントリの延べ数、使用されているエントリ数の変化、エントリの最大使用数について計測した。

(2) 一括GCによる輸出入表の回収

一括GCでは輸入表を探索して参照されていない輸入表エントリを回収し、それが指す輸出表に対してreleaseメッセージを送出する。一括GCによる輸出入表の回収がどのくらい起きているかを調べるために、一括GCを起こす状況で各エントリの使用数の変化を計測した。

(3) 分割輸出の頻度

データの参照カウントに重みを付けたことによって、分割輸出する場合には輸出表を操作しなくても良い。この効果を調べるために、総輸出回数に対する分割輸出の割合を計測した。

(4) unifyメッセージの削減率

2.6で述べた、変数セルと外部参照セルのユニフィケーションにおけるunifyメッセージ送出を抑える効果を調べるために、変数セルと外部参照セルの各組合せの頻度を計測した。

3.4 計測結果および評価

3.4.1 輸出入表のインクリメンタルな回収

図7は輸出入表の使用状況をグラフにしたものである。プログラムはPentominoで、クラスタ数4のうちの0番クラスタについてのものである。他のクラスタについてもほぼ同じ変化が見られた。プログラムが終了するまでの間には一度もクラスタ内の一括GCが起きていない。図中の輸出表・輸入表総割付数は割り付けられたエントリの積算値をプロットしたものである。輸出表・輸入表使用数はその時実際に使用されていたエントリの数をプロットしたものである。左側が単一参照の輸出入表で、右側が多重参照のものについて調べたグラフである。

輸出表、輸入表ともに、総割付数が大きく増加しているのに対して、使用数はあまり増加していない。これはインクリメンタルな回収によるものである。

表3に各プログラムにおける輸出入表エントリの総割付数と最大使用数をまとめた。それぞれ全クラスタの合計値である。測定値の半分は最大使用数の割合が10%未満である。30%を越えるものは1割しかない。このことにより、「インクリメンタルな回収によって輸出入表を効率的に使用している。」と言える。

3.4.2 一括GCによる輸出入表の回収

図8はシミュレータのヒープメモリの大きさを小さくして、プログラム実行中に一括GCを起こした場合のデータをグラフにしたものである。縦軸は実際に使用していた輸出入表エントリの数であり、横軸はリダクション数である。プログラムはPaxで、2クラスタで実行した時の0番クラスタにおけるものである。

55Kと59Kリダクション付近で各エントリ数が急激に減少しているが、0番クラスタでは55Kリダクションに1番クラスタでは59Kリダクションに一括GCが起きてい

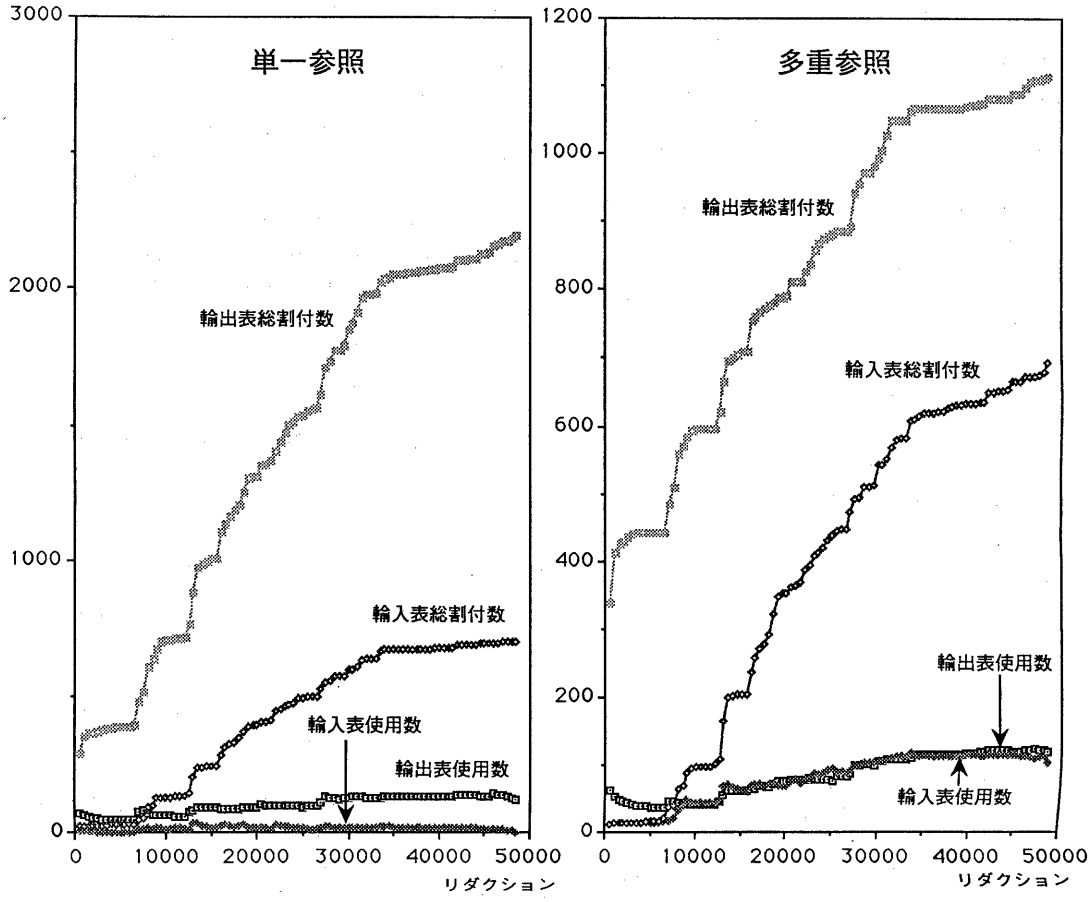

図 7: 輸出入表の使用状況 (Pentomino 4cluster)

る。55K リダクションにおける輸入表エントリの減少は自クラスタの一括 GC による回収を、59K リダクションにおける輸出表エントリの減少は、他クラスタの一括 GC によって送出された release メッセージを受信し、その結果輸出表が回収されたことを意味する。

3.4.3 WEC の効果

表 4 に分割輸出の頻度を示す。総輸出回数との比較を見ると、Tsumego 以外のプログラムではかなり高い値を示している。また、クラスタ台数が増すにつれて割合が増している。2.5節で述べたように、分割輸出の割合が高いことにより WEC の効果は大きい。

3.4.4 unify メッセージの削減

表 5 は外部参照セルと変数セルのユニフィケーションが生じた時点で、そのうち unify メッセージを送出したものの割合をまとめたものである。2.6節で述べた、2 つの最適化についてそれを入れた時にどのように変化したかを示す。最適化は、(1) 輸出済み変数セルであることを示すタイプの導入、(2) 構造体を指す外部参照ポインタであることを示すタイプの導入である。(1) と (2) 両方を入れた場合は、Bestpath 以外のプログラムでは unify メッセージ送出を 0 %にできた。Bestpath においても 50 %であったのを 20 %に削減できている。Tsumego ですべて 0 %になっているのは、外部参照セルが全て Safe 属性であったためである。また、今回の計測では (1) のみの最適化で十分であるような結果が出ている。今後より多くのプログラムでデータを収集して、もっと詳しい検討が必要である。

表 4: 分割輸出の頻度

	2 CL	4 CL	6 CL
Pentomino	1560(75.7)	2906(87.9)	3566(90.2)
Bestpath	6332(63.1)	9525(83.7)	12094(88.7)
Pax	31315(57.2)	64636(69.2)	78555(73.1)
Tsumego	173(8.5)	366(26.2)	408(27.4)

() 内は総輸出回数に対する割合 (%)

表 3: 輸出入表の総割付数と最大使用数

	Pentomino			Bestpath		
総割付数	2 CL	4 CL	6 CL	2 CL	4 CL	6 CL
単一外部参照 輸出	5256	7978	9086	35865	39556	44591
輸入	1957	2696	2906	21593	26840	30299
多重外部参照 輸出	2060	3307	3952	10029	11385	13628
輸入	1814	2942	3552	7091	10780	14082
最大使用数	2 CL	4 CL	6 CL	2 CL	4 CL	6 CL
単一外部参照 輸出	363(6.9)	508(6.4)	581(6.4)	4156(11.6)	5112(12.9)	6086(13.6)
輸入	65(3.3)	116(4.3)	145(5.0)	2444(11.3)	2811(10.5)	3293(10.9)
多重外部参照 輸出	327(15.9)	435(13.2)	468(11.8)	2155(21.5)	3259(28.6)	4239(31.1)
輸入	330(18.2)	507(17.2)	612(17.2)	2157(30.4)	3724(34.5)	5004(35.5)

	Pax			Tumego		
総割り付け数	2 CL	4 CL	6 CL	2 CL	4 CL	6 CL
単一外部参照 輸出	62597	104512	123586	4919	2977	2807
輸入	6930	7354	7434	2382	1315	1177
多重外部参照 輸出	54718	93367	107437	1935	1395	1491
輸入	54376	104836	126953	1536	1242	1402
最大使用数	2 CL	4 CL	6 CL	2 CL	4 CL	6 CL
単一外部参照 輸出	3699(5.9)	4271(4.1)	4467(3.6)	389(7.9)	254(8.5)	229(8.2)
輸入	162(2.3)	184(2.5)	229(3.1)	241(10.1)	86(6.5)	87(7.4)
多重外部参照 輸出	3553(6.5)	4125(4.4)	4283(4.0)	167(8.6)	156(11.2)	145(9.7)
輸入	3550(6.5)	7479(7.1)	9193(7.2)	173(11.3)	346(27.9)	528(37.7)

() 内の数は総割付数に対する最大使用数の割合 (%)

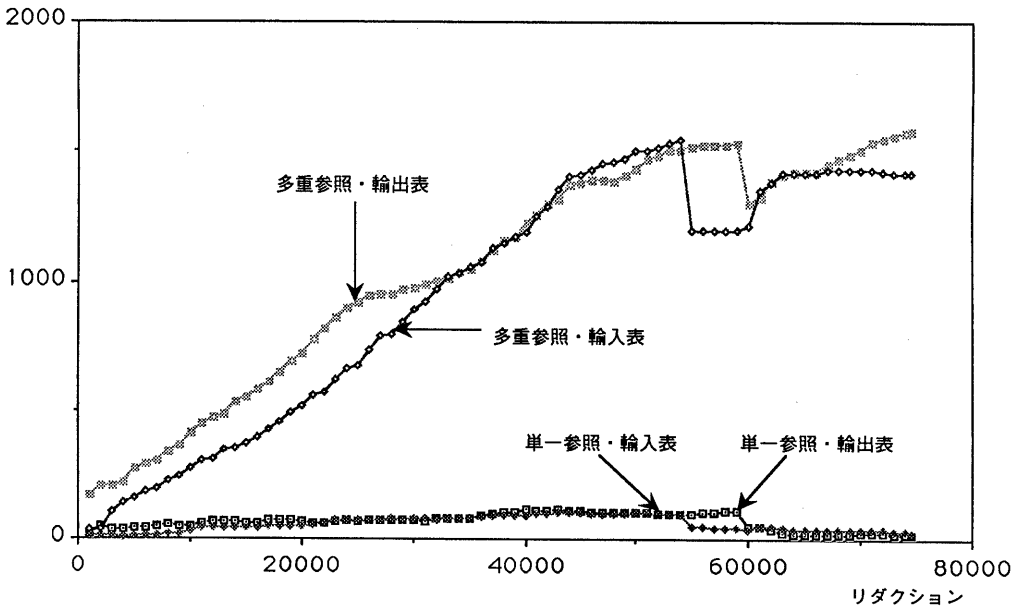

図 8: 一括 GC による輸出入表の変化

表 5: 全ユニフィケーションに対する unify メッセージを送出するものの割合 (%)

	Pentomino			Bestpath			Pax			Tsumego		
クラスタ数	2	4	6	2	4	6	2	4	6	2	4	6
最適化なし	38.5	49.2	57.3	39.9	43.6	43.7	50.5	45.9	35.0	0	0	0
(1) 輸出済み変数セル	0	0	0	20.0	17.3	16.6	0	0	0	0	0	0
(2) 構造体を指す外部参照ポインタ	38.5	49.2	48.0	39.8	43.6	43.7	0	0	0	0	0	0
(1) と (2) 両方	0	0	0	20.0	17.3	16.6	0	0	0	0	0	0

4 おわりに

VPIM を C 言語に変換して作成したシミュレータにより、KL1 処理系のクラスタ間に渡るデータの参照方式の評価を行なった。今回の評価では以下にあげる処理方式上の有効性が確認された。

- 輸出入表エントリは、インクリメンタルな回収によって、総割付数に対する最大使用数の割合が 10 %程度に抑えられた。
- 一括 GC 時の release メッセージ送出により輸出表エントリの回収があった。
- 総輸出に対する分割輸出の割合が高いことから、WEC 方式による効果が大きい。
- 変数セルと外部参照セルのユニフィケーションでは、最適化により最大 50 %の unify メッセージが抑えられた。

我々がクラスタ間データ処理方式の評価において目標とするものは、そのシステム全体から見た性能 (処理速度、処理コスト) である。今回得た評価結果は、その手がかりとなるものである。シミュレータで収集できるデータは、まだ多く残されている。今後、さらなるデータの収集によって VPIM におけるクラスタ間データ処理方式の評価を続ける必要がある。

謝辞

日頃ご指導頂いている瀧和男室長をはじめとする ICOT 第 1 研究室の研究員の方々に感謝致します。また、ベンチマークプログラムを提供して下さったプログラム開発グループの皆様に感謝致します。

参考文献

1) A.Goto, et.al., "Overview of the Parallel Inference Machine Architecture (PIM)", In Proc. of the International Conference On FGCS'88, pp208-229, 1988.

2) K. Ueda and T. Chikayama., "Design of the Kernal Language for the Parallel Inference Machine", The Computer Journal, 33(6), 1990.

3) 平野 他, 「並列論理型言語 KL1 のコンパイル方式の改良」, 並列処理シンポジウム JSPP'90, 1990.

4) 山本 他, 「並列推論マシン PIM における抽象機械語 KL1-B の実装 - 高級機械語を実装するための道具立 -」, 信学技報 CPSY 89(168), 1989.

5) N.Ichiyoshi, et.al., "A New External Reference Management and Distributed Unfication for KL1", In Proc. of the International Conference on Fifth Generation Computer Systems, ICOT, Tokyo, 1988.

6) 六沢 他, 「分散環境におけるユニフィケーションの実現」, 第 37 回情処全大 7Y-4, 1988.