PRAMプログラムからPVMプログラムへの変換*

1L－6

金山 二郎　飯塚 肇†
成蹊大学大学院工学研究科‡

1 はじめに

近年、並列計算の研究開発が盛んであり、独自のプログラミング方式を備えた様々なアーキテクチャが存在している。そのような状況において、アーキテクチャに依存しない、汎用的な並列プログラム記述手法の開発が切望されている。

この問題に対して、仮想並列計算機PRAMをプログラミングパラダイムとした言語を設計し、各種並列計算機に実装することによる解決を試みる。

PRAMは共有メモリと共通クロックを持つ仮想並列計算機であり、同期命令を要さない。また、複数のメモリモデルが用意されており、問題に適切なモデルを選択することで、並列プログラムを簡潔に記述することが可能である。

本稿では、PRAMプログラムから分散型並列計算環境のプログラムへの変換可能性について考察し、実現の一部として、PVMライブラリが用意されている計算機上での実装を試みる。

2 環境

2.1 PRAM

PRAM(Parallel Ramdom Access Machine)は、共通クロックを持つ共有メモリ型の仮想並列計算機である(図1)。主に並列アルゴリズムを抽象的に考察するために広く用いられ、並列性を簡潔に記述できる。ただし、プロセッサ数に上限がないなどの理由から、実装は不可能である。

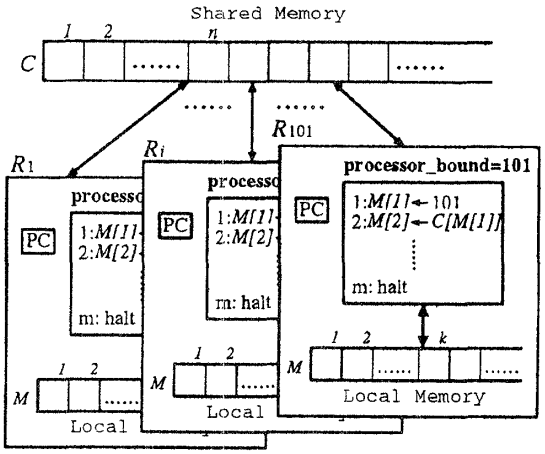

図 1: PRAM

PRAMの各プロセッサはRAM(Random Access Machine)である。これは、局所メモリを持つ一般的なプロセッサを模倣している。各マシン語命令は同じクロック時間を消費し、共通クロックに従って実行される。

メモリモデルは、読込と書込について、同時発生を許すか否かが選択できる。また、プログラムは各プロセッサで一斉に動作開始し、全てのプログラムが正常終了した時点でPRAMの動作が正常終了したものとみなす。初期データと結果のデータは、共有メモリに格納されていると仮定している。

2.2 PRAMアルゴリズム

プログラム記述を簡潔かつ明瞭にするために、PRAMアセンブリ言語に替わってPRAMアルゴリズムが用いられることが多い。これはAdaなどの構造化言語に似ており、プロセッサ数の指定や並列実行文など、PRAM特有の機能をサポートしている。本稿では、これをPRAMプログラムとして扱う。

PRAMアルゴリズムの定義は文献によって若干異なるが、本稿では[3]のものを用いる。

2.3 PVMライブラリ

PVMライブラリは、分散環境において複数プロセスがデータを送受信するメッセージパッシング機能を提供する。

特徴として、歴史的な経緯と汎用性から、メッセージパッシングライブラリとしては最も広く用いられており、ネットワーク結合されたワークステーションや著名な並列計算機のほとんどすべてに実装されている。

なお本稿では、ethernetで結合されたSunワークステーション8台で構成された分散環境を想定した。

3 実装方針

方針として、1) 並列化と2) 大量データ処理を考える。2)に関しては、並列プログラムの多くが非常に大量のデータを扱うために、その抽象化を考慮すべきだという判断から挙げたが、本稿では割愛する。

並列化の単位　PRAMアルゴリズムにおいて明示的な並列実行部を表すpar文の並列実行を考える。par文は、「集合Xの各要素pに対して、任意のプロセッサに$OPERATION(p)$を割り当てて実行させる」という意味を持つ。

$$\textbf{par}\ p \in X\ \textbf{do}\ OPERATION(p)$$

データ分割数は、基本的には実際の計算機のプロセッサ数で割る。このpar文が終了するまで、次のpar文の実行はブロックされる。これにより、一時に存在するプロセ

*Translation from PRAM program to PVM program
†Jiro KANAYAMA and Hajime IIZUKA
‡Seikei University

スの数を実際のプロセッサ数以下に抑える。また、このことにより、プログラムの意味の保持を par 文の単位に分割する。par 文の入れ子については、最初の par 文で生成されたプロセスの範疇で実行される。

統治管理 ある親プロセスを仮定し、そのプロセスが必要に応じた数のプロセスを生成させる。プログラムは SPMD(Single Program Multiple Data) の形式をとる。共有メモリの初期的な入力と最終的な出力も、このプロセスが属する計算機上のハードディスク上にファイルとして配置される。

データ分割 1) 計算機の数 2) データ量 3) データ依存性を判断基準にデータ分割を行う。最終的には、4) ネットワークの形態 5) 計算機/ネットワークの性能も加味する。

プログラム変換 データ分割に伴い、各プロセスで行われる処理を決定する。通信を発生させて並列プログラムとしての補完を行うが、基本的には通信回数を最小限に抑える形での変換がなされる。

4 並列化の例—リストランキング—

リストランキングのプログラムを図 2 に示す。大文字で始まる変数は共有変数を表す。

```
processsor_bound = n;
par(i: 1<=i<=n){
    par(j: Key[i]<=j<=maxkey){
        key_density[j]++;
    }
}
for(i=1;i<=n;i++){
    Rank[i] = --key_density[Key[i]];
}
```

図 2: リストランキングのプログラム

Key[] にはあらかじめランキングの対象とされるキーが格納され、実際のランク付けは Rank[] に格納される。maxkey は最大のキー値であり、key_density[0..maxkey] の各要素 key_density[n] には、n 以下であるようなキーの総数が格納される。

入出力データの分割/収集 まず、Key[],Rank[] は p 個のプロセスに均等に分割配置される。分割配置は親プロセスによりなされ、Rank[] も親プロセスに集められてファイルとして格納される。

キーデンシティの計算 通信回数を最小限に抑えるというポリシーの下で、key_density[] は一旦、各プロセスに配置された Key[] の部分リストについてなされる。その後、隣のプロセスに伝播する形で統合され、最終的な結果がブロードキャストされる (図 3)。

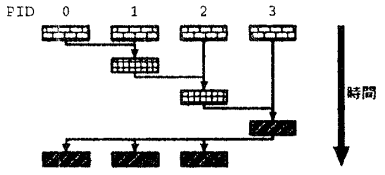

図 3: key_density[] の統合の様子

ランキングの計算 最終的に得られた key_density[] と、それを伝播する際に用いられた一時的なキーデンシティを利用して、各プロセス内の Key[] の部分リストについてランキングを計算する。

性能評価 生成された PVM プログラムの性能を、1) 逐次版, 2)TreadMark 版[1] と比較する (図 4)。両プログラムは、いずれもキーデンシティを計算する形でのリストランキングを行う。また、TreadMark 版では key_density[] が共有配列としてデーモンにより管理される。Key[] として int 型のデータを 2^{20}元用意し、同じソートを 10 回行わせた。

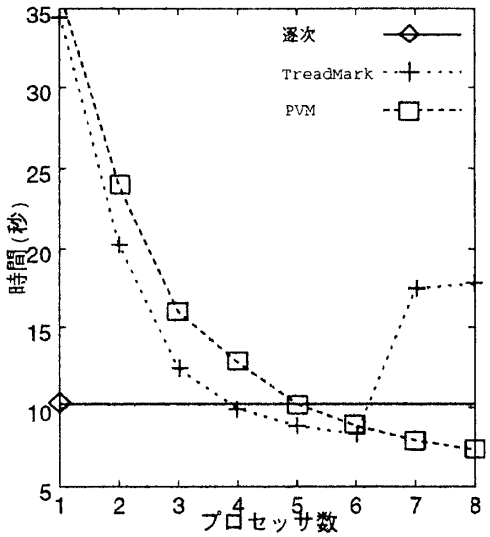

図 4: 性能の比較

結果として、6 台程度から効果が見られた。TreadMark 版が素直な台数効果を発揮しないのは、TreadMark の構造上の問題であると考えられる。変換された PVM プログラムが TreadMark 版のものに対してそれほどひけをとらなかったことから、性能としても一定の水準を満たしていると考える。

5 おわりに

PRAM アルゴリズムを用いて、PVM プログラムの生成を行った。PVM プログラムを直接記述しないことで、並列プログラミングの難度を緩和した。また、性能的にも一定の水準を満たしていることを立証した。

データ分散/収集/統合アルゴリズムについて、現在では線形な方式を採用しているが、プロセッサ数が数十のオーダであった場合、性能的な欠陥となる。より汎用性の高いアルゴリズムへの変換方法を模索している。また、メモリ量が制約された条件下でのプログラム変換についても検討している。

参考文献

[1] Karp R.K, et.al, Efficient PRAM Simulation on a Distributed Memory Machine, Algorithmica(1996)

[2] 金山二郎, 飯塚肇, *PRAM* プログラムから *pthread* プログラムへの変換, 情報処理学会第 52 回全国大会 (1996)

[3] 宮野悟, 並列アルゴリズム, 近代科学社 (1993)

[1] DSM(Distributed Shared Memory) の一種