7F−3

並行記述言語NCCのMach上への実装と評価*

徳吉 隆宏　木村 哲郎　天野 英晴†

慶應義塾大学理工学部‡

1 はじめに

並列計算機の普及につれ、多くの並行記述言語が提案されている。本大学でも、科学技術計算を対象とした並行記述用の言語としてNCC [1] を提案している。

NCCはNCモデル [4] と呼ばれる静的な並行プロセスモデルに基づく並行記述言語であり、マルチプロセッサ上での科学技術計算を効率的に行うための必要最小限の構成を持つ。NCCでは並行プロセス記述能力を制限することにより、実行時処理系のオーバヘッドを軽減し、また高い解析性および可読性をもつ科学技術計算用プログラムを記述することができる。

現在NCCは本大学で開発されたバス結合型並列計算機 Attempt-0 [2] 上に実装されている。Attempt-0ではDIPROS [3] というNCC専用の処理系が稼働している。DIPROSはスケジューリングや通信管理を行う一種のOSであり、他の並列計算機への移植は困難である。従ってNCCの移植も難しく、使用できる環境は限られている。

そこで本研究では、DIPROSのような専用の処理系に依存しない、Machを搭載した汎用並列計算機へのNCCの実装法を提案する。この実装法は、多くの並列計算機でサポートされているC Threadsライブラリに基づいており、より多くの並列計算機への移植が可能となる。

2 並行プロセス記述言語NCC

NCCはC言語をベースとする言語である。NCCはメッセージパッシングに基づく言語であり、メッセージはコネクティングラインと呼ばれる通信路を介して伝送される。NCCでは、対象とする問題はプロセスとコネクティングラインで記述される。プロセスはポートを介してコネクティングラインと接続しており、このポートを介して他のプロセスとの通信を行なう。

NCCは以下の特徴を持つ。

- ネットワークは静的に張られる。またプロセスも静的であり、動的な生成消滅はない。
- 通信のためのバッファは1語である。
- 1対多の通信を行うマルチキャストが可能である。
- 通常の受信の他に複数の入力ポートに対して非決定的受信を行うことができる。これにより複数のメッセージを到着順に処理をする機能を備えている。

*An Implementation and An Evaluation of NCC on Mach

†Takahiro TOKUYOSHI, Tetsurou KIMURA, Hideharu AMANO

‡Keio University

3 Mach上への実装

3.1 実装方針

NCCの実装における要点は、メッセージ通信の支援とプロセス・スケジューリングである。特に、メッセージ通信に伴うオーバヘッドが処理速度に大きな影響を及ぼす。そこでMach上への実装では、NCCのプロセスをスレッドに対応させ、スレッド間で共有されるデータ領域を利用して、メッセージ通信のオーバヘッドの低減を図る。共有変数を用いてメッセージ通信を実現するためには、1語のバッファを持つ通信に必要な同期操作が必要となる。ここでは、C Threadsで提供されている同期プリミティブを用いてメッセージ通信の支援を行う。この同期プリミティブの利用により、スレッドのスケジューリングはMachが担当することになる。

NCCはC言語をベースとした言語であり、そのプログラムは明示的に宣言されたプロセスの集まりとして記述されている。これをthread fork等を用いたスレッドの動的生成の形に置換し、交信に関する情報をプログラム中に挿入することにより、C Threadsのプログラムに変換することが可能である。本実装は、NCCのプログラムをC Threadsのプログラムに変換するトランスレータと、送受信文のためのライブラリで構成される。

3.2 通信用ライブラリ

この通信用ライブラリはNCCの通信機構をC Threadsライブラリを用いて実現した部分である。

NCCはメッセージパッシング型の言語である。これに対して、C Threadsでの通信は共有変数を用いて実現されている。よって共有変数を用いて、メッセージパッシング型の通信を実現する必要がある。

また、マルチキャストは1対多の通信なので、それに関わるプロセスが多くなり、処理が繁雑になりがちである。この問題をよりシンプルに扱うためにPark and Bench Model(PBM) が提案されている。本実装もこのPBMに基づいている。

PBMではプロセス間の関係、即ち通信路の状態が一括して扱われ、ポートが通信可能であるのかどうかが、容易に判定できる。

PBMでのメッセージの送受信はparkとbenchで行なわれる。送信メッセージはparkに送られ、benchを介して受信される。1つのparkに対して複数のbenchを対応させると、マルチキャストとなる。図1にparkとbenchの関係の摸式図を示す。

C Threadsを用いたライブラリでは、parkとbenchへのアクセスの際にロックを掛けてPBMを実現し

てある。送信を試みて、それが不可能であった場合スレッドはブロックする。他のスレッドの通信により、ブロックしたスレッドの通信が可能になった時、再びその通信が試みられる。図2に受信文の書式、図3に受信文に対応するライブラリを示す。

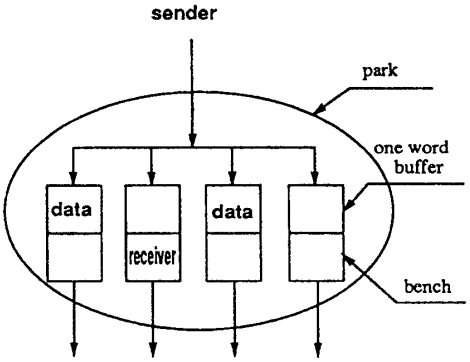

図 1: park と bench の関係

3.3 トランスレータ

このトランスレータは、NCC のソースを変換するプログラムである。このトランスレータにより、通信に必要な情報を得るためのコードがNCC のプログラムに埋め込まれる。

変換後の NCC は main 関数と そこからフォークして実行される関数からなる。ここで言うフォークとはプロセスのフォークではなく、スレッドのフォークである。

main 関数はスレッド間の共有情報の初期設定をしてフォークを行う。この共有情報は通信のためのもので、コンパイル時に解析され、中間ファイルとして生成される。

呼び出される関数では、通信に必要な共有情報を読み込み、そのスレッドのコードを実行する。スレッドのプログラムは共有情報を読み込む部分だけが変換され、残りの部分は変換されない。

4 おわりに

本研究で提案した方法に基づいて実装されたNCC の実行時間を表1に示す。この評価は OMRON/LUNA-88K 上で、3つのプログラムについて、PU 数を変化させてとったものである。3つのプログラムは、連立方程式を解くもの (jacobi)、行列の固有値を求めるもの (eigen value)、素数を求めるもの (eratosthenes) である。この表は、PU 数の増加に対し、性能の向上は頭うちであることを示している。これはコンテキストスイッチのオーバヘッドのためであり、特にスレッド数が多い時に深刻な問題となる。

このように、台数効果の点で若干の問題はあるが本研究の実装法により、一般の汎用並列計算機へのNCC の実装が可能となった。

表 1: PU 数に対する実行時間 (単位:秒)

application	1PU	2PU	3PU	4PU
jacobi(17threads)	29.0	19.3	13.4	12.7
eigen value(17threads)	55.5	32.8	29.1	27.3
eratosthenes(101threads)	92.0	55.2	52.3	49.8

receive(入力ポート名, 受信変数のアドレス);

図 2:NCC の受信文の書式

```
receive(port_id,variable)
int     port_id ;
double  *variable;
{
  mutex_lock(& park.lock);

  if(park.status == EMPTY){
    park.status = RECWAIT;
    /* Wait */
    condition_wait(& park.arrive_data, & park.lock);
  }

  /* Can Receive data */
  if( park.status == SENDWAIT){

    park.status = EMPTY;
    *variable = data;  /* Receive data */
    park.count--;

    if(park.count == 0 ){
        /* Activate Sender */
        condition_signal(& park.no_data );
     }
  }else{  /* Error */
        error();
  }
  mutex_unlock(& park.lock);
}
```

図 3:受信のためのライブラリ

参考文献

[1] 朴 泰佑, 工藤 知宏, 天野 英晴, 木村 哲郎, "マルチプロセッサのための科学技術計算用並行記述言語 NCC", 電子情報通信学会論文誌 D-I Vol.J72-D-I No.10 pp.750-758 1989 年 10 月

[2] 鳥居 淳, 天野英晴, "並列計算機テストベッド ATTEMPT の交信機構の評価", 並列処理シンポジウム JSPP '91 論文集, pp.205-212, 1991

[3] T.Boku, T.Kudoh, H.Amano, H.Aiso, "DIPROS - A distributed processing system for NDL on $(SM)^2$ -II," Proc. of the 12th HAWAII International Conferrence on System Sciences, vol.2, pp.208-217, 1987

[4] T.Kudoh,H.Amano,T.Boku,H.Aiso,"NDL: A language for solving scientific problems on MIMD machines," Proc. of the 1st Super Computing Symposium,1985.