並列計算機EM-4の並列プログラミング言語EM-C

6M−1

佐藤三久 児玉祐悦 坂井修一 山口喜教
電子技術総合研究所

1 はじめに

EM-4[1]は、高速なデータ駆動機構をもつ分散メモリ型の並列計算機である。本稿では、EM-4の並列プログラミングのために開発したC言語のsupersetであるEM-Cについて述べる。

EM-4のデータ駆動機構は、データフローパケットによるプロセッサ間の通信によって、それに対応するスレッドを高速に起動・同期することを可能にしている。プロセッサはネットワークに対してメッセージを直接送りだしたり、受けとったメッセージに対するスレッドの起動をハードウエアの機構としてサポートしている。EM-Cでは、データ駆動機構を逐次実行を行なうスレッドの起動あるいは同期の機構として用いる。このプログラミングモデルを、データフロー計算機本来のデータフローモデルに対して、マルチスレッドプログラミングモデルと呼んでいる[2]。このモデルでは、EM-4はプロセッサ間でのスレッド生成やリモートプロシージャコールが非常に早いプロセッサと見ることができる。

EM-4では、このデータ駆動機構をデータフロー実行だけでなく、遠隔のメモリの読みだし・書き込みにも拡張している。EM-4は、基本的には分散メモリのマルチプロセッサであり、各プロセッサはローカルなメモリをもち、グローバルな共有メモリはもたない。しかし、メッセージはシステムで定義されている特定のスレッドを実行させるようにすることができる。この機能を用いて、他のプロプロセッサのメモリに対してアクセスすることが可能である。EM-Cはリモートメモリアクセスの機能を用いて、ユーザに対して、仮想的な共有メモリを提供している。すべてのプロセッサのローカルメモリをアドレスづけできる空間であり、これを分散グローバルアドレス空間(distributed global address space)と呼んでいる。プログラマは、この空間上にデータを分散配置し、共有メモリとして直接アクセスすることができる。

EM-Cは、EM-4 nativeな言語として、プログラマに対して、EM-4における並列プログラムの構造的な記述を提供することを目的としている。EM-Cは並列記述として、以下のものを提供している。

- スレッド生成やメッセージ通信などの基本的な並列操作のコンパイラ組み込みオペレーション
- 分散グローバルアドレス空間でのデータの分散配置とアクセス
- direct matching機構で同期するための並列化構文

2 コンパイラ組み込み並列オペレーション

スレッドの生成 —スレッドは、fork操作によって生成することができる。

```
fork(pe_addr,func,arg1,....)
```

pe_addrで示されるPEに関数funcを実行するスレッドを生成する。生成されたスレッドは、関数funcの実行が終了した時に自動的に消滅する。

同期 —通常、スレッドはその実行を終了する前に他のスレッドに対して、同期操作を用いて、実行の終了を通知する。同期メモリ操作では、I_readは、I_writeで書き込まれるまで、実行がブロックされる。また、ライブラリとして、バリア同期が用意されている。

Remote call — forkのかわりにremote_callを用いることによって、元のスレッドは、関数の実行が終了するまで待たされる。これは指定されたPEにおいて関数を実行するRemote Callになる。

メッセージ通信 —スレッドに対してメッセージ通信を行なう場合は、create_thread操作を用いて、スレッドを生成する。

```
port=create_thread(pe_addr,func,arg,...)
```

返されたportに対して、word_sendを用いてメッセージを送ることができる。また、word_recvで、現在実行中のスレッドに送られてきたメッセージを受け取ることができる。

Operation # Arguments	Time (μs) 0	1	2	3
local call/fork	1.68	2.16	2.64	3.12
remote fork	2.10	2.51	2.98	3.46
remote call	4.75	5.16	5.64	6.12
create thread	6.24	6.64	7.13	7.60

表1: Thread Operation Time on EM-4

Operation	Time (μs)
remote word read	1.85
remote word write	0.40
ping word msg	5.66
ping packet	3.71

表2: Communication Operation Time on EM-4

EM-C: A Parallel Programming Language for The EM-4 Multiprocessor
SATO Mitsuhisa, KODAMA Yuetu, SAKAI Shuichi and YAMAGUCHI Yoshinori
Electrotechnical Laboratory

表1にEM-Cでの基本操作の実行時間を示す。リモートな操作は、80PEに対する平均実行時間である。remote callは、空な関数を実行した。表2に基本的な通信操作の時間を示す。ping word msgは、メッセージ通信によるword転送を各PE間で往復させる時間である。ping packetは、1 word bufferで済む場合の最適化した場合である。なお、EM-4は、12.5MHzのクロックで動作しており、メモリ参照命令などを除く大部分の命令は1クロックサイクルで実行される。ネットワークの性能はPEのポート当たり60.9 Mbytes/secである。

3 グローバルアドレス空間とデータ分散

同じPEで実行されるスレッドは、同じローカルメモリ空間を共有する。プログラム中で宣言されるデータは、各プロセッサ上のローカルメモリの同一アドレスに配置される。EM-Cでは、配列データの宣言に対して、globalキーワードを付け加えることによって、プログラマは分散グローバルアドレス空間上に配列などのデータを分散配置し、直接アクセスすることができる。コンパイラは、グローバルポインタの参照に関して、リモートメモリアクセスのコードを生成する。また、ローカルなポインタとの代入操作で、グローバルなアドレスを明示的に作り出して、他のPEで参照することができる。

分散グローバルアドレスを用いることによって、共有メモリ並列プログラムを実現することが容易になる。その一つの例として、SPLASHベンチマークの並列スタンダードセルルータLocusRouteをEM-4上に実現したが、共有メモリプロセッサと同程度の性能向上をすることを示されている[3]。

4 並列化構文

EM-4では、データフローのdirect matchingを使うことによって、効率的な同期が可能である。構文を拡張し、並列化構文を導入することによって、direct matchingで同期できるようにした。

parallel構文 — *parallel*構文は、複文内の各文を並列実行するcobegin/coend型の構文である。並列実行の同期は、direct matching機構を使って行なわれる。

タスクブロック: *Forkwith* と *Dowith* — *forkwith*構文は、一連のコード（これをタスクブロックと呼ぶ）を実行するスレッドを生成する。

`forkwith(`*parameters*`){` *task body ...*`}`

パラメータリストとして、タスクブロックから参照する変数を指定する。これらの変数は、タスク生成時にコピーされる。*forkwith*の代わりに*dowith*を使うことによって、元のスレッドはタスクブロックが終了するまで、ブロックされる。

*iterate*構文を用いることによって、同じタスクブロックを複数生成することができる。ループなどを並列実行し、他のプロセッサに対するリモートメモリアクセスやリモートオペレーションのレーテンシをプロセッサ内の並列性で隠蔽し、効率化するために用いる。

Where構文 — *where*構文は、データのアドレスによって、スレッドを制御する。

`where(`*global address*`)` *task block statement*

指定されたグローバルアドレスで指定されるPEでタスクブロックを実行する。

*everywhere*構文は、すべてのPEに同じタスクブロックを実行するスレッドを生成する。配列など分散グローバルアドレス空間に分散配置されたデータ構造に対しては*everywhere*構文を使って、各プロセッサに同一なスレッドを生成し、データ並列操作などを容易にプログラミングできる。*dowith*で、同期することによって、バリア同期としても使うことができる。スレッドの生成はtree上に行なわれ、*dowith*のタスクブロックの同期を必要とする場合はデータフローのマッチングを使って行なわれる。ちなみに現在のインプリメントでは、空のタスクブロックに対して、69μsecで実行できる。

5 おわりに

EM-Cでは、データ駆動機構を高速にスレッドを制御・同期する機構と捉えなおすことによって、従来のプログラミングモデルとの連続性のあるスレッドを基本としたプログラミングを行なう。スレッドの制御やメッセージ通信、分散グローバルアドレスでのリモートメモリアクセスは、すべてEM-4のデータ駆動機構の枠組内で効率的に実現されている。

この言語の目的は、EM-4での並列プログラミングのための構造的な基盤を与えることにある。並列化構文で記述された並列プログラムを最適化することにより、データフローモデルで得られるコードに極めて近いコードを生成することができる。EM-Cはすでに、データ並列言語などのバックエンドとして使われており、さらにhigh levelなSISALなどの関数型言語をEM-Cの上に実現することを計画している。

謝辞 本研究を遂行するにあたり御指導、御討論いただいた弓場情報アーキテクチャ部長、計算機方式研究室の同僚諸氏に感謝いたします。

参考文献

[1] S. Sakai, Y. Yamaguti, K. Hiraki, Y. Kodama and T. Yuba, "An Architecture of a Dataflow Single Chip Processor", *Proc. of the 16th Annual International Symposium on Computer Architecture*, pp. 46-53, June 1989.

[2] M. Sato, Y. Kodama, S. Sakai, Y. Yamaguchi and Y. Koumuara, "Thread-based Programming for the EM-4 Hybrid Dataflow Machine", *Proc. of of the 19th Annual International Symposium on Computer Architecture*, pp. 146-155, May 1992.

[3] 佐藤、児玉、坂井、山口、並列計算機EM-4上での共有メモリベンチマークの実行, 信学技報、CPSY 92-61, pp. 49-56, 1992.