5C-7

関数レベルデータ駆動計算機の検討

戸田賢二　内堀義信　弓場敏嗣
(電子技術総合研究所)

1．はじめに

汎用・高性能計算機の実現を考えた場合、マルチプロセッサ方式による高並列計算機がアーキテクチャ側の結論であろう。各要素プロセッサ（ＰＥ）に、いかに低コストでＰＥの使用効率を増すように負荷を分散するかというのが課題である。

2．データ駆動方式の問題点

データ駆動方式は、プロセッサの制御を意識することなくデータを転送することだけで負荷分散が行えるため、マルチプロセッサシステムとの整合性が高い。しかし、命令レベルでデータの待ち合せを行い、他ＰＥへの負荷分散が行える状況を作りだしていることは、逐次処理方式と比較して次の問題点を招いている（ここでは汎用性のあるタグ付きトークンによるダイナミックアーキテクチャを仮定する）。

（１）ハードウェア量が多い
データの待ち合せを行う連想メモリが必要であり、連鎖ハッシュ方式を採用している科学技術計算用データ駆動計算機 SIGMA-1【１】では通常メモリの１．８倍のメモリ量を要している。また、データにはタグが付いているため通信関係の処理部分のバンド幅が増加する。このため命令のアドレス空間が圧迫されたりする。 SIGMA-1ではデータ部分とタグ部分でバスをマルチプレックス使用している。

（２）関数呼び出しが遅く、数に制限がある
同一関数が複数のものから同時に呼ばれるため、それらの識別子（カラー）が必要である。このカラーの割り付け、戻り先へのリンク作成、実引数と仮引数のバインディング等が必要であり、ダイナミックリンク方式のSIGMA-1 では同一ＰＥ内での関数起動で“８＋２×引数個”の命令を要する。これは関数呼び出しを他ＰＥへ送出するための機構を実現しているためであるが、ＰＥ内で頻繁に関数呼び出しが起こる状況では大きなオーバーヘッドとなる。カラーはデータのタグとして持ち歩くためそのサイズや、戻り先へのリンクを保持するレジスタ数等による制限がある。

（３）書き替え可能な記憶要素が必要である
データ待ち合せの機能だけでは、大規模データの部分更新や履歴保存等の作業が非効率であるため、書き替え可能な記憶要素（連想メモリを高機能化したり、別にストラクチャメモリ等を付加して実現する）が必要となる。これは、データ駆動原理を一部崩すことになり、本来のデータ依存性を考慮するだけでよいという利点を損なうことになる。また、ｎオペランド命令においては命令発火速度は、平均的に連想記憶の処理速度の１／ｎとなるので高速化のために連想記憶にデータを張り付けたり、命令実行部にレジスタを設けたりするチューニングがあるが、これらも同様である。

（４）スタンドアローン化が困難
ＰＥ数の増加やＰＥの複雑化等によってハードウエアコストが大きくなる程、マルチユーザ・マルチタスクの要求は高まるが、タスク管理機能・ハードウェア資源管理機能等はデータ駆動方式だけで実現するには困難がある【２】。勿論、データ駆動計算機はその中に逐次処理プロセッサを持ち、プログラム実行の一部やデータ駆動部分の保守・管理を行うものであってもよい。問題は、それらの整合性がよくシステムとして効率的に動作するかどうかという点である。純粋なデータ駆動プロセッサに逐次処理プロセッサを併用するのはハードウェアの無駄で、２つが融合できなければならないと考える。

3．何故関数レベルか？

データ駆動制御を命令レベルで行わずもっと大きな単位で行うマクロデータフローの試みには、Cedar【３】がある。これは、大規模マルチプロセッサシステムを対象として、コンパイラによって命令のデータ依存性を解析し、それによってまとめられた命令群を単位とするものである。

これに対し、我々は関数呼び出しはプログラミングの必須概念であり、関数のモジュラリティや再帰性を多用したプログラミングが自然であるとの考え方に立つ。また、関数というものに対象を限定することで効率的な実現が可能となる。並列性の抽出についても、我々が開発した記号処理用データ駆動計算機ＥＭ－３（ＰＥはマイクロコンピュータベースでデータ駆動プロセッサをシミュレートする）の評価結果【４】等から、関数単位で十分な負荷分散が行えることが示されている。これらのことから、汎用・高性能計算機として以下に示す関数レベルデータ駆動計算機を提案する。

4．動作方式

本方式の基本原理は次の通りである。

（１）関数呼び出しを含め処理はできるだけＰＥ内で行い、逐次処理をする。
（２）仕事の分配は要求駆動で関数単位とし、関数の返値の待ち合せを行う。

これに基づいて、関数呼び出しの形で与えられるタスクに対するＰＥの動作を図１に示す。

返値の待ち合せ機能については、値待ち時間と値待ち合せのためのタスクの実行中断・再開に要するオーバヘッド時間を減少させるために次のものを導入する。

（１）単一待ち合せ
値参照を値設定時まで待たせる。
（２）ＡＮＹ待ち合せ
複数の値参照候補のうち１つでも参照可能となるまで待たせる。値はそのもの。
（３）ＡＮＤ（ＯＲ）待ち合せ
複数の値参照候補についてＡＮＤ（ＯＲ）条件が決定されるまで待たせる。値はＡＮＤ（ＯＲ）の結果。

5．言語

本方式に適合するプログラミング言語は次の特徴を持つものである。

A Function Level Data Flow Computer
Kenji Toda, Yoshinobu Uchibori, Toshitsugu Yuba
Electrotechnical Laboratory

（１）負荷分散の観点から関数呼び出しの頻度がある程度以上あるようなプログラミングスタイルが要求されるので、関数呼び出しのオーバヘッドが小さく再帰呼び出し可能であること。
（２）関数内では、逐次実行処理方式の利点を生かして副作用を持った演算（ループ制御等）の記述が可能なこと。
（３）関数外へ副作用を及ぼすことがないこと。
（４）これらの制限を持つ分配可能関数の他、自己実行用関数の定義ができ、そこでは関数外部への副作用を持つ演算も可能であること。
（５）関数は複数返値とすること。

以上の性質から、手続き型言語で純粋な関数の記述が容易なＣ言語に以下の機能を追加したものを基本言語とする。

（１）関数呼び出し
　　f (arg) ――――自己実行
　　f <arg>――――分配可能
分配可能呼び出しは実行時に、他ＰＥからのタスク要求があるかどうかで動的に分配が決まる。外部に副作用を与える関数は、自己実行呼び出ししかできない。
（２）関数値の特殊待ち合せ演算子
　　ＡＮＹ待ち合せ―――@
ＡＮＤ、ＯＲについてはＣ言語の論理演算子と同様。
（３）関数の引数、返値とも任意サイズ
```
a [] = f <args>;
v1, v2, ... , vn = f <args>;
return (w0, w1, ... , b [4-10] ) ;
```
式の値は返値の最初のワードとする。

６．　ハードウェア

本方式は、均一ＰＥをネットワークで結合したマルチプロセッサシステムをベースにしたものであり、以下のハードウェア機能により効率的に実現される。

（１）ネットワークの機能
タスクを要求駆動で分散するため、アイドリングＰＥを随時知らせる機能があること。
（２）ＰＥの機能
１）プロセッサ
逐次型であり、実行環境切り換えの機構の効率的なサポートができるもので、アドレス空間の切り換え及びその保護が行えるメモリマネージメント機能が必要である。

①中断していたタスクのうち実行可能となっているものがあれば、それを再開する（③へ行く）。

②新たに実行できるタスクがあれば、それを実行する。なければ、他のＰＥにタスク要求を出す（①へ行く）。

③◇タスク実行中に関数呼び出しに出会ったときは他のＰＥからのタスク要求があれば、戻り先をつけてタスクとして分配する。要求がなければ、自分で逐次的に実行する。

◇タスク実行中に分配したタスクの返値に対する待ち合せが発生したら、実行を中断する（①へ行く）。

◇タスクが終了すれば、戻り先にその関数値を返す（①へ行く）。

図１．ＰＥの動作方式

２）待ち合せ機能付きメモリ
関数の返値の待ち合せを効率的に行うため、図２の機能を持ったメモリが必要である。関数が他のＰＥに分配された場合、返値の待ち合せ条件に応じてこれらのメモリを初期設定する。返値の参照はこのメモリを読むだけでよく、関数のリターンは指定されたメモリに値を書き込むだけでよい。

７．　今後の予定

現在、本方式についてＥＭ－３のハードウェア(MC68000×１６＋ルータネットワーク）をテストベッド【５】として用いて評価を進めている。これでは、タスク要求をパケット転送で行い、待ち合せメモリの部分をソフトウェアでシミュレーションしている。
今後、この上で応用プログラムを作成し、問題の記述性、並列性の抽出具合等について調べ本方式の実現可能性を評価したい。

謝辞
柏木寛電子計算機部長、計算機方式研究室の同僚諸氏及び神戸製鋼所の大塚喜久氏に感謝致します。

参考文献
【１】Hiraki, K. et al. ,"A Hardware Design of the SIGMA-1 — A Data Flow Computer for Scientific Computations," Proc. Int. Conf. Parallel Processing, IEEE, 1984
【２】戸田: データ駆動計算機のスタンドアローン化における問題点の検討，第３２回情処全大，6R-7，1986
【３】Gajski, D., D. Kuck, D. Lawrie and A. Sameh, "Cedar—A Large Scale Multiprocessor," Proc. Int. Conf. Parallel Processing, IEEE, 1983
【４】Toda, K., Y. Yamaguchi, Y. Uchibori and T. Yuba, "Preliminary Measurements of the ETL LISP-based Data-driven Machine," FGCA(IFIP TC-10), 1985
【５】内堀，戸田，弓場: 並列処理用テストベッドの試作，第３２回情処全大，5Q-9，1986

単一待ち合せ・ＡＮＹ待ち合せ用メモリ
初期値（Ｒ，Ｗはフラグ）
　Ｒ←０，Ｗ←０
メモリ参照
　Ｗ＝１――通常の内容読み出し
　Ｗ＝０――トラップ発生(read trap)　Ｒ←１
メモリ書き込み
　Ｒ＝０――通常の書き込み　　Ｗ←１
　Ｒ＝１――書き込み後トラップ発生
　　(write trap)　　Ｒ←０，Ｗ←１

ＡＮＤ待ち合せ・ＯＲ待ち合せ用メモリ
初期値（Ｒ，Ｍはフラグ、Ｃはカウンタ）
　Ｒ←０，Ｃ←待ち合せ数，Ｍ←０(ＡＮＤ)
　　１(ＯＲ)
メモリ参照
　Ｃ＝０――通常の内容読み出し
　Ｃ≠０――トラップ発生(read trap)　Ｒ←１
メモリ書き込み
　Ｒ＝０――書き込み内容の論理値が
　　Ｍならば　　Ｃ←０
　　そうでなければ　　Ｃ←Ｃ－１
　Ｒ＝１――Ｒ＝０の場合と同様、その後トラップ発生 (write trap)　Ｒ←０

図２．特殊待ち合せ用メモリ