5C-5

並列処理における動的負荷分散方式 －帳尻合わせ方式－

内堀義信　戸田賢二
（電子技術総合研究所）

1．はじめに

先に報告した並列処理用テストベッド[1]（６８０００×１６台、ＰＥ間の通信は主としてルータネットワークによる）の上で並列処理を効率よく行うことができる動的負荷分散方式－帳尻合わせ方式－を提案し、その実行制御機構と予備評価について述べる。

2．背景

一つのジョブを個別に実行可能な単位（本稿ではこれをプロセスという）に分割し、各プロセスを適当に要素プロセッサ（ＰＥ）に割り当て、多数台の要素プロセッサが協調して並列に処理する並列計算機においては、各ＰＥ（要素プロセッサ）の効率が上がるように負荷を配分することが重要である。全負荷はすべてのプロセスの実行時間の総和として求められる。

負荷分散方式は、プログラムコンパイル時に行う静的負荷分散と、実行時に行う動的負荷分散に大きく分けられる。すべての処理について言えることであるが、コンパイル時に処理することができれば実行時オーバヘッドをなくすことができるため静的負荷分散方式の方が望ましいのは言うまでもない。

しかしそのためには、プロセスの実行回数があらかじめ推定でき、かつその数が比較的少なく、しかも各プロセスの実行時間が推定できなければならない。ところが、一般的に、各プロセスの実行時間を実行する前に知ることは困難である。また、条件文や再起呼び出し等が存在するため、プロセスの実行回数をあらかじめ知ることも困難である。これらの理由により、静的にプロセスを割り当てる方法、プロセスの個数のみを評価基準にして動的負荷分散を行おうとする方法には限界がある。

並列処理におけるジョブの実行時間は最後のプロセスが終了した時点によって規定される。これは次のことを意味する。Ｎ台のＰＥで並列に処理することを考える。Ｎ－１台のＰＥの負荷はまったく等しく、１台のＰＥの負荷だけが倍になった場合を考えると、実行時間は理想値のほぼ２倍になる（Ｎ＝１６の場合１．８８倍である）。逆に、１台のＰＥの負荷がゼロになった場合を考えると、実行時間は理想値とほとんど変わらない（Ｎ＝１６の場合でも６．７％増加するだけである）。つまり、並列処理において重要なのは、特定のＰＥに負荷が集中してピークが生じるのを避けることであり、必ずしも負荷が均等に配分される必要はないということである。

3．前提条件

本稿で提案する負荷分散方式ではＰＥ間の通信にルータネットワークのみを用いており、その他の特別なハードウェアは不要である。

対象プログラムに関して必須となる条件及び望ましい条件は以下の通りである。

必須条件

各プロセスはＰＥ依存性がなく、他のＰＥへの再配置が可能であること

望ましい条件

各プロセスは関数性が高く、プロセス間の干渉がないこと

本稿の議論では特に断わらない限り上記の条件がすべて満たされている場合を想定している。

4．基本方針

各プロセスの実行時間はそのプロセスの実行が終了した時点で確定するため、いくつかのプロセスの実行が終了した時点で、負荷の再配分をすることが効果的であると考えられる。つまり、ジョブの後半部で各ＰＥに残っているプロセスを再配分しながら処理を進めれば、ほぼ理想的な負荷分散が行えるはずである。これが本方式の基本方針であり、命名の由来でもある。

並列処理においてはＰＥ間の通信がネックになり易いため、プロセスの生成は必要最小限にすることが望ましい。この観点から言えば要求駆動型の負荷分散方式が最も望ましいことになる。本稿の方式もジョブの起動時を除けば、一見要求駆動型のように振舞う。

ＰＥ当りのプロセスの個数が比較的少ない場合やプロセスの生成を抑制するメカニズムが組み込まれている場合は、乱数的にプロセスを分散させるよりも一定の順序ですべてのＰＥを回るように配分した方がよい。

Dynamic load balancing -Post balancing method-

Yoshinobu UCHIBORI, Kenji TODA
Electro-technical Laboratory

5．実行制御機構

前項までの考察から、一様配分機構、動的再配置機構、プロセス生成抑制機構の３つを備えた実行制御機構（以下ではスケジューラと呼ぶ）を試作した。

ジョブが起動されてから終了するまでの、各ＰＥにおけるスケジューラの実行シーケンスは次のようになる。

１）プログラムがロードされて、初期化が正常に行われると２）へ行く。

２）スケジューラはプロセス待ち状態に入る。プロセス生成抑止フラグを降ろす。ＣＰ（制御プロセッサ）か他のＰＥからプロセスを受け取ると３）へ行く。

３）プロセス生成モードの実行状態に入り、プロセス生成カウントが設定値を越えるまでサブプロセスを生成しながら実行する。プロセス生成カウントが設定値を越えたら、カウントを０に戻し、４）へ行く。プロセスキューが空になったら５）へ行く。

４）プロセス生成抑止フラグを立てる。サブプロセスの生成は止まり、実行はセルフコールで行われる。この状態で、他のＰＥからＥｍｐｔｙパケットを受け取り、かつそのＰＥにプロセスを分け与えることが出来なかった場合には３）に戻る。プロセスキューが空になったら５）へ行く。

５）Ｅｍｐｔｙパケットを送出し、アイドル状態に入る。プロセスパケットを受け取ったら、プロセス生成抑止フラグに応じて３）または４）へ。自分の出したＥｍｐｔｙパケットを受け取ったら２）へ行く。

図．１はスケジューラの状態遷移図である。

6．予備評価

本方式の有効性を検証するために、Ｎ－ＱＵＥＥＮ問題（Ｑ（Ｎ））を例に取り上げ、予備評価を行った。この実験では、プロセス生成抑制については、疑似的に対象プログラムの中で行っている。

これらのメカニズムはＣで記述されており、対象プログラムもＣで書かれている。コンパイラは既存のものをそのまま用いているため、並列実行部分はファンクションコールによって陽に指定しなければならない。

Ｑ（８）からＱ（１２）に対してＰＥ台数を１台から１６台まで変えて処理時間を実測した結果を表．１に示す。性能向上曲線は図．２の様になる。

計時は内蔵タイマーを用いて、ＰＥ０を起動してからすべてのＰＥが停止するまでの時間を測定した。

図に示されている通り、Ｑ（１２）ではほぼ理想的な性能向上が得られている。これは、元々Ｑ（Ｎ）問題の性質が良いことと、本方式のオーバヘッドが十分小さいことによる。Ｑ（８）の性能向上がそれほど芳しくないのは、ジョブの立ち上がり・終了時のオーバヘッドの他、各ＰＥのスケジューラの状態変化をＣＰ（制御プロセッサ）に送って、それを表示しているため、ＰＥ台数が増えるとＣＰ－ＰＥ間の通信量が増えるためと思われる。

7．おわりに

予備評価により、本方式が原理的に有効なことが明らかになった。今後、前述の機構を全て実装してより詳細な評価・検討を行う予定である。

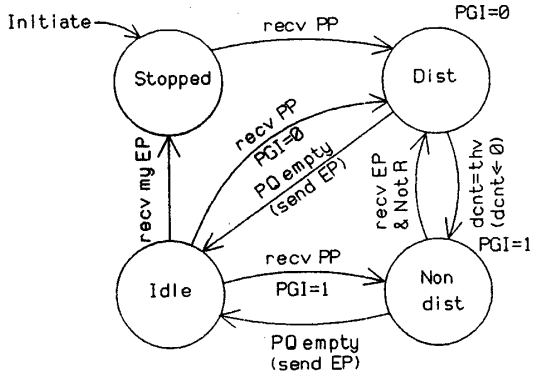
謝辞　本研究の機会を与えて頂いた柏木電子計算機部長、弓場計算機方式研究室長ならびに日頃ご討論頂く計算機方式研究室の同僚諸氏に感謝します。

参考文献

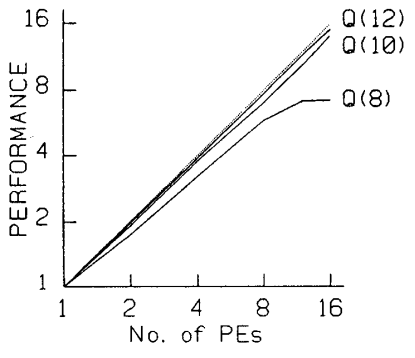
［１］内堀、戸田、弓場：並列処理研究用テストベッドの試作、第３２回情処全大、５Ｑ－９（１９８６）

ＰＥ台数	Ｑ（８）	Ｑ（１０）	Ｑ（１２）
1	1．13	28．75	974．2
2	0．64	14．82	491．2
4	0．35	7．53	246．1
8	0．19	4．0	123．8
12	0．16	2．7	83．5
16	0．16	2．02	64．4

表．１　Ｑ（Ｎ）計算時間（単位は秒）



図．１　スケジューラの状態遷移



図．２　Ｑ（Ｎ）の性能向上曲線