2J－6

相互接続された異種計算機環境における並列計算のための負荷分散について

久野 英治　渋沢 進　小林 正樹
職業能力開発大学校情報工学科　茨城大学工学部情報工学科

1　はじめに

近年、プロセッサ及び周辺機器の性能向上が、パーソナルコンピュータと言う名称を意味のないものにしつつある。一方、データの共有と計算能力の分散を可能とするネットワークの重要性が高まり、共通のプロトコルによって構築されるネットワークは、今やあらゆる計算機をも接続し、共通のサービスをそれら計算機に提供している。このようなネットワークとそこに接続された複数の計算機とを仮想的なマルチプロセッサシステムと見なすソフトウェア PVM(Parallel Virtual Machine) [1] が供給され、それを用いたアプリケーション開発が各所で盛んに行なわれている。

本稿では、PVM を用いた負荷分散手法について提案する。まず PVM の概要を述べ、次にその環境におけるシステムの負荷と見なせるものを提示し、次にそれらを検討し、その結果を元にして負荷分散手法を提案する。

2　PVM の提供する環境

PVM は、TCP/IP をベースとしたネットワークとそこに接続された UNIX OS を持つ計算機からなる環境であれば、マルチプロセッサシステムを構築することが可能であると言われている [2]。しかし、ネットワークの性能の問題により、通信のコストが高い。そのため、PVM では、メッセージ交換方式によるの疎結合型のマルチプロセッサシステムモデルを採用している。

接続された計算機の性能はファミリー系統のプロセッサであれば、単なる相対値で容易されており、計算量との積によって、負荷ないし計算量を分散を考えることが可能である。しかし、異なる設計思想のプロセッサもネットワークに接続されているため、それらの相対性能もデータベーステーブル上に用意されている。

PVM では、UNIX におけるプロセスを一つの仮想的な要素プロセッサと見なすため、場合によっては n プロセス/プロセッサ ($n \geq 1$) という構成も考えなければならない。この場合の負荷は、UNIX のプロセス毎に管理することになり、UNIX 側の負荷との兼ね合いを考慮しなければならない。

3　負荷の定義

PVM 動作時における負荷の種類とその内容について分類する。

システム負荷 — OS 側の視点からの負荷で、必ずしも計算余力だけを表現しているわけではない。

計算量 — プログラムの計算量を分割してある程度の機能単位毎にまとめて、手続きのシーケンスとして細分化したものである。

トラフィック — ネットワーク上を通過する単位時間当たりのデータ量である。

ただし、ここではシステム外部からのトラフィックや負荷が全くないものと仮定している。

4　負荷分散の検討

PVM では、UNIX のプロセスを要素プロセッサとしているために、システム負荷と言う観点から見ると、単体のプロセスの場合、システム負荷との比は 1:1 となるが、必ずしも他の負荷の単位と一致するとは限らない。また、複数の要素プロセッサを割り当てる場合、それぞれの負荷とシステム負荷のミスマッチが大きくなる。しかし、実計算機のシステム負荷が 1 に近づきさらに増加する傾向を示すときにはターゲット先として選ぶのは意味がない。

A Load Sharing for Parallel Computation in a Heterogeneous Computing Environment
KUNO, Eiji
University of Industrial Technology
4-1-1, Hashimoto-Dai, Sagamihara, Kanagawa 229, Japan
SIBUSAWA, Susumu
KOBAYASHI, Masaki
Ibaraki University
4-12-1, Naka-Narusawa, Hitachi, Ibaraki 316, Japan

次に、計算量の観点から見た場合、要素プロセッサが保持する処理単位数を負荷とすると、それぞれの計算量が見積もれる時には、役立つ。しかし、見積れないと、最悪の場合一つの要素プロセッサに大量の計算量を持つ処理単位が割当てられる可能性がある。

トラフィックの観点からだと、トラフィックが高いと言うことは、処理単位間でのネットワークを介した通信が頻発していると判断できる。その一方で、長いパケット転送の可能性もある。ここでの判断基準はTCP/IPにおける応答パケットのタイムアウトなのでそちらに依存することとする。タイムアウトになった場合は、負荷に相当するデータは送らないこととする。

5 負荷分散の手法

ここで仮定するプログラムモデルは、任意の機能単位毎に処理がまとめられたuプロセス[1]という並列処理単位と、それらをなんらかの順序関係(以下、シーケンスと呼称)で結んだ構成をとっている。そして、特定のuプロセス間では通信や同期に相当する処理が行なわれる。

基本方針としては、オーバヘッドの少ないプログラム実行と高速な実行である。そこでシーケンスの早いuプロセスは先に実行させる。また、生産者/消費者問題のようなuプロセスが存在した場合は、できるだけ生産者側の先行させる。前者は、uプロセスが早期に多数存在させて負荷の分散を済ませておくことを意味する。後者は消費者によるデータ待ちを出来るだけ避けさせて、一種のコンテキスト切替と同等の処理を回避させることを意味する。

以上の点を考慮して、uプロセスにはプロセス識別子とシーケンス識別子を保持させる。初期値を1として起動時のuプロセスに与え、子が生成されて親と子のようなシーケンスが必要になった時に、親のシーケンス識別子をカウントアップした値を子に渡すもの(図1の pid$i$ と pid$i+1$ 参照)とする。ただし、生成される子のuプロセスが多数存在し、それらが同格の場合は、子と同じシーケンス番号をコピーして渡す(図1の $\mathrm{pid}i''_n$ (n = 1, 2, ...) 参照)。以上の形式を採用する。制御については、シーケンス番号が同じものについては各計算機への分散の対象とすることにより、シーケンス番号が同じuプロセスは分散させることとする。次に、本来の利用方法として、シーケンス番号の小さいものを優先して実行させる。これによって、できるだけ待合わせに伴う処理のオーバヘッドを回避する。

この処置での負荷の定義は、要素プロセッサが保持するuプロセスのシーケンス番号の総和を保持するuプロセス数で割った値とする。その値が大きければ大きいほど処理が進行しているものと仮定できる。注意しなければならないのは、待ち合わせに入ったuプロセスを要素プロセッサの負荷の算出対象としないことである。また、平均値が0になる仮想プロセッサは負荷が無いと解釈し、負荷分散のターゲットとする。

次に、同期処理が行なわれる場合、同期対象となったuプロセスのなかで最も大きいシーケンス番号を対象となった全てのuプロセスにコピーして修正する。これによって、同期後の処理がほぼ同程度の時間で各uプロセスの処理が開始できる。

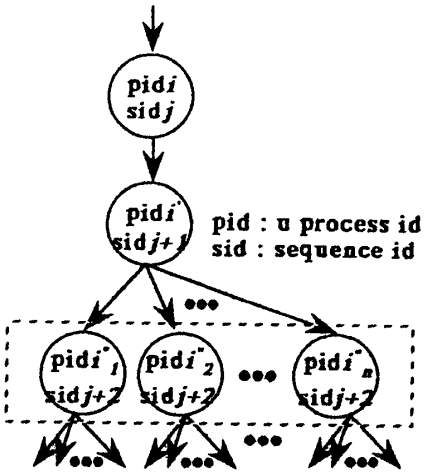

図 1: シーケンス id の設定例

6 おわりに

3つの負荷をベースにした、PVMを用いた負荷分散手法について提案した。今後は、実アプリケーションを用いてこの機構の評価を行なっていく予定である。

参考文献

[1] A. Beguelin, J. J. Dongarra, G. A. Geist, and V. S. Sunderam *"Visualization and Debugging in a Heterogeneous Environment"*, IEEE Computer, pp.88 – 95, June 1993

[2] A. Beguelin, J. J. Dongarra, G. A. Geist, R. Manchek and V. S. Sunderam *"A Users' Guide to PVM parallel Virtual Machine"*, Technical Report ORNL/TM-11826, Oak Ridge National Laboratory, July 1991.

[1] UNIX と異なることを示すために u を加えて呼称する。