1G－7

モジュール分割を用いたプログラムの並列化の効果について

菊地重昭（東北工大）　白鳥則郎（東北大通研）　宮崎正俊（東北大）

1．はじめに

Fortran 語等の逐次型高水準言語を用いたプログラムを並列化して処理効率を上げるための手法として、プログラムをモジュールに分割し、モジュール単位で並列処理する並列化手法がある[1]。ここでは、この並列化手法の効果を評価するために、この並列化手法を適用したシステムのモデルを提示し、いくつかの仮定を設定した後、待ち行列理論を用いてシステム解析する。解析結果を数値計算用ライブラリ IMSL について、この並列化手法を適用した場合と適用しない場合について具体的な値として求め、手法の効果を評価する。解析に際して必要となる IMSL に関する以下に示すような統計的諸量についてもここで述べる。

(1) モジュールの実行時間の分布。

(2) 並列処理できるモジュールの数、すなわち、並列度の分布。

(3) プログラムにおける並列に処理できるモジュールの実行を1単位とした時の実行段階数、すなわち、ステップ数の分布。（並列化した場合としない場合の両者について）

2．システムモデルとシステム解析のための仮定

システムモデルを図1に示す。

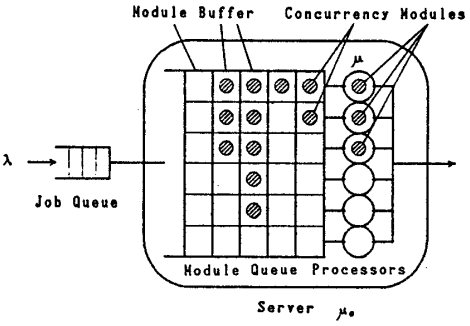

図1．システムモデル

次に、このシステムのシステム解析に当っての仮定を以下に上げる。

[仮定]

（a）システムに到着するプログラムは、たがいに独立なプログラムである。

（b）システムへのプログラムの到着は、平均λのポアッソン到着に従う。

（c）システムへのプログラムの到着を受入れるジョブキューは、棄却のない無限長のキューである。

（d）サーバには、同一時刻内では、ただ一つのプログラムしか滞在できない。

（e）サーバ内の並列化モジュールキューは、棄却のない無限長のキューである。

（f）並列化モジュールキュー内のモジュールバッファは無限個のモジュールを受入れることの出来るバッファである。

（g）並列プロセッサは、たがいに独立に動作する無限個のプロセッサからなる。

（h）並列化モジュールキュー内の先頭のモジュールバッファにあるモジュールは、並列プロセッサ内に未処理のモジュールが全くない時のみ並列プロセッサに送り込れ並列処理される。

（i）モジュール内の並列処理は、しないものとする。

（j）ジョブキューから並列化モジュールキューに送り込まれる時のモジュール分割や並列性の検出、並列化等に要するオーバーヘッドは、無視出来るものとする。

3．モジュールに関する統計的諸量

数値計算用ライブラリー IMSL にモジュール分割を用いた並列化手法を適用した時、1の(1)、(2)、(3) に記述した統計的諸量について述べる。

（1）モジュールの実行時間の分布

IMSLの中の9本のプログラムにつき、モジュール分割した後の184モジュールにつきコンピュータ上での実行時間を測定した。この結果から実行時間の確率分布曲

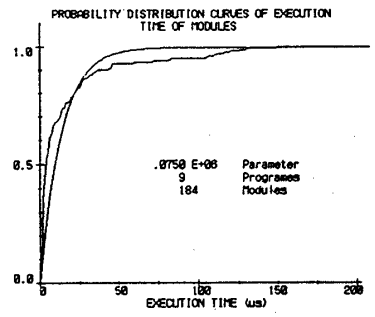

図2　モジュールの実行時間の確率分布

Evaluation with Parallel Processing System Using the Concept of Module Partition

Shigeaki KIKUCHI, Tohoku Institute of Technology
Norio SHIRATORI, Reseach Institute of Electrical Communication, Tohoku University
Masatoshi MIYAZAKI, College of General Education, Tohoku University

線を求めたところ、図2のようにパラメータμ=75000の指数分布とみなせる。

（2）並列度の分布

３６１本のプログラムの実行開始点から終了点までについて調査した結果、そこに含まれる９７９２個のモジュールの並列度の分布は図3に示すように p_p=0.771 の幾何分布 $F_p=\sum_{k=1}^{\infty} p_p(1-p_p)^{i-1}$ とみなせる。

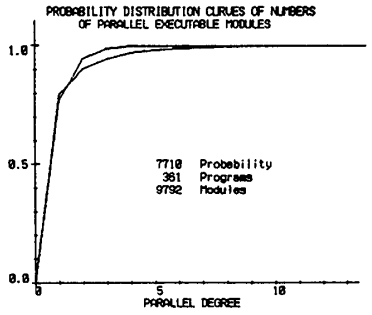

図3　モジュールの並列度の分布

（3）実行ステップ数の分布

３６１本のプログラムについて、プログラムの実行開始点から終了点までの任意のパスを選び並列化した後のパスにつき実行ステップ数を調べた結果は図4の通り。

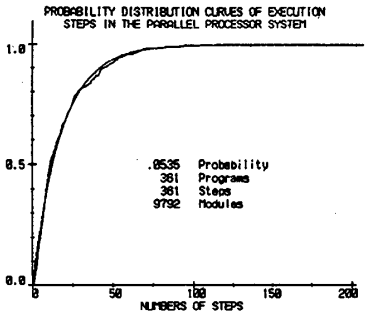

図4　実行ステップ数の分布

図より実行ステップ数の分布は、p_s=0.0535 の幾何分布で $F_s=\sum_{k=1}^{\infty} p_s(1-p_s)^{i-1}$ とみなせる。また、並列化しない場合は、p_s=0.0325 幾何分布とみなせる。

４．システム解析と並列化手法の評価

このシステムを M/G/1型待ち行列システムとして、平均システム内滞在時間と平均待ち時間を求める。このため、図1のサーバのサービス時間の平均値 $1/\mu_0$ およびその2次モーメント b_2 を求めると式(1)、(2)のようになる。

$$1/\mu_0 = -\log(p_p)/(p_s(1-p_p)) \quad (1)$$

$$b_2 = 2(1-p_s)/\mu_0^2 + (2/\mu^2)\cdot\left(\sum_{k=1}^{\infty}(1/n)\cdot\left(\sum_{m=1}^{40}((1-p_p)^{m-1}/m)\right)\right) \quad (2)$$

$$1/\mu_0 = 1/(p_s\mu) \quad (3)$$

$$b_2 = 2(1-p_s)/\mu_0^2 \quad (4)$$

M/G/1 型待ち行列システムの平均システム内滞在時間 W_q および平均待ち時間 Wは、式(5)、(6)のようになることが知られている。ただし、ここで $\rho=\lambda/\mu_0$ である。

$$W_q = (\lambda^2 b_2+2\rho-2\rho^2)/(2(1-\rho)) \quad (5)$$

$$W = b_2/(2(1-\rho)) \quad (6)$$

以上のことから並列化した場合と、しない場合の平均システム内滞在時間 W_q、W_q'を到着プログラム数　に対して求め図示すると図5のようになる。図中には、実測データを用いてコンピュータシミュレーションをした結果のデータも記述されている。

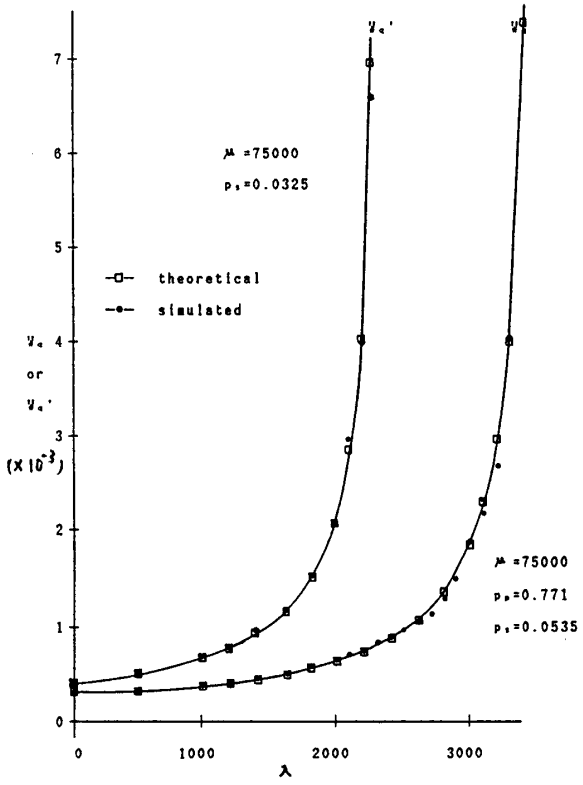

図5　平均システム内滞在時間

５．むすび

図5からモジュール分割を用いた並列化手法の効果をみると、約2倍程度改善されていることがわかる。これに加えて、変数に名前をつけるに当り、モジュール間の変数依存関係をできるだけなくす等、並列化の最適化を導入すれば、より効果が上ることが期待できる。

６．参考文献

(1) 菊地、白鳥、宮崎「逐次型高水準言語プログラムのモジュール分割による並列性の抽出について」電情通学論 D, Aug.'88

(2) 菊地、白鳥、宮崎「モジュール分割によるプログラムの並列性の解