オペレーティング・システム　46－4
（１９９０．　２．　２３）

モジュール分割による並列化プログラムの並列性の解析

菊地　重昭　　白鳥　則郎　　宮崎　正俊
（東北工大）　（東北大通研）　（東北大）

筆者等は、FORTRAN語等の逐次型高水準言語によるプログラムの並列化手法として、主として、プログラム中の制御文を境にして、モジュールと称するサブプログラムに分割し、プログラムを全体にわたりモジュール単位で並列化する手法を提案している。ここでは、この手法による並列化の効果を評価する。このため、待ち行列理論に基づいた解析手法により、並列化の効果を定量的に解析する方法を提案する。そして、本解析手法の適用例として、代表的な数値計算用ライブラリ中のプログラムに適用した場合について述べる。このとき、これらのプログラムをモジュール分割した場合の統計データを導入し、プログラムの平均システム内滞在時間を求めてモジュール分割による並列化手法の適用効果について評価する。また、シミュレーションによる解析結果を求め、本解析手法による結果と比較する。

EVALUATION OF PARALLEL PROCESSING OF A PROGRAM USING THE CONCEPT OF MODULE PARTITION

Shigeaki Kikuchi[1]　　Norio Shiratori[2]　　Masatoshi Miyazaki[3]

Fuculty of Engineering, Tohoku Institute of Technology[1]
Research Institute of Electrical Communication, Tohoku University[2]
College of General Education, Tohoku University[3]

35-1 Yagiyama Kasumicho, Taihaku-Ku, Sendai-Shi, 982 Japan[1]
2-1-1 Katahira, Aoba-Ku, Sendai-Shi, 980 Japan[2]
Kawauchi, Aoba-Ku, Sendai-shi, 980 Japan[3]

Several interesting parallel processing methods have been proposed in the recent past. Some of them deal mainly with the parallelism of the individual program module such as DO-statement in a Fortran program, while a few of others are rarely concerned with the parallelism of the whole of the large-scale programs. In this paper, we discuss evaluation of efficiency of parallel processing for the large-scale program which is divided into four types of program modules in the whole of the program, in addition to using the loop parallelism. This parallel processing method can achieve an improvement of average total time spent in the system. For example in subroutine programs of IMSL which is a typical numerical calculation library, the average response time by our method is smaller than three quarters of the value by using only the loop parallelism method.

1　まえがき

並列処理の研究には、言語レベルのものとアーキテクチュアレベルのもの等がある。言語レベルの並列処理の研究の中に、FORTRAN 語等の逐次型高水準言語によるプログラムを並列化する研究がある。しかし、それ等の多くは、DO文のような繰返し処理の内部を並列化するための研究[1]~[6] であるが、並列化を繰返し処理にのみ適用するのではなく、プログラム全体にわたって適用する研究[7]~[10]がある。文献(8) では、モジュール分割という再帰的に適用できる分割手法を提案している。これは、プログラムをモジュールと呼ばれるサブプログラムに分割した後、モジュール内の並列性だけでなくモジュール間の並列性も検出し、プログラムを全体にわたって並列化する並列化手法である。

ここでは、モジュール分割を用いた並列化手法の並列化の効果を評価するため、この並列化手法を適用したシステムについて待ち行列理論に基づいた解析手法によりシステム解析し、プログラムの平均システム内滞在時間を求めて、並列処理しない場合の値と比較し並列化の効果を調べた。解析に際しては、以下の統計的諸量を導入している。

（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の分布、すなわち並列度の分布
（3）並列に処理できるモジュールの実行を１単位とした時のプログラム当りのステップ数の分布
（4）並列処理しない場合のプログラム当りのステップ数の分布

解析の適用例として代表的な数値計算用ライブラリ、IMSLを用いている。また、システムの解析結果を検証するために計算機シミュレーションによりプログラムの平均システム内滞在時間を求めた。

2　並列化手法の概要とシステムモデル

2．1　モジュール分割を用いた並列化の概要

FORTRAN 語等の逐次型高水準言語で記述されたプログラムについて、プログラムを並列処理の可能性を持つサブプログラムに分割するモジュール分割法を導入する。このため、プログラムを構成している実行文をデータ処理を目的とするデータ処理文と、データ処理文の実行順序の制御を目的とするフロー制御文の２種類に分ける。

データ処理文：　代入文、READ文、WRITE文
フロー制御文：　DO文、IF THEN文、ELSE IF文、
　　ELSE文、END IF文、CALL文、
　　RETURN文、STOP文

次に、並列化するプログラムについて、主としてフロー制御文に注目し、以下の４つの種類のモジュールと称するサブプログラムに分割する。

（1）タイプB(ブランチ)のモジュール
　IF文から END IF文までのサブプログラム
（2）タイプC(コール)のモジュール
　プログラム中 CALL 文で呼び出されるサブルーチン副プログラム
（3）タイプD(データ処理)のモジュール
　連続した一つ以上のデータ処理文の集まり
（4）タイプL(ループ)のモジュール
　DOループにおいて、DO文から端末文までのサブプログラム

タイプLのモジュールの並列化については、ここでは従来の並列化手法を用いることとし、モジュール内の分割は行なわないものとする。それ以外のモジュールについては、モジュールの内部についても、この分割手法を適用できなくなるまで再帰的に適用する。モジュール分割が終わったら、隣接するモジュール間でモジュール内の変数依存関係を調べ、依存関係のないモジュールは並列処理可能なモジュールとし並列化する。タイプLのモジュールについては、DOループの繰返し処理に対する従来の並列化手法を適用する。

2．2　システムモデル

モジュール分割手法を用いた並列処理システムのシステムモデルを図１に示す。

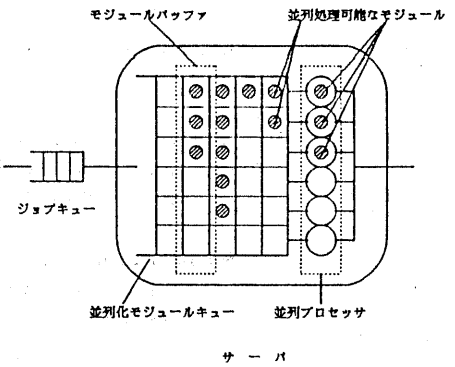

図１　システムモデル

システムは、ジョブキューとサーバからなり、以下の仮定のもとで動作するものとする。

<<仮定>>
（1）プログラムの到着は、平均入のポアッソン到着。
（2）プログラムは、たがいに独立である。
（3）ジョブキューは、無限長のキューである。
（4）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。
（5）サーバ内の並列化モジュールキューは、無限長のキューである。
（6）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できるバッファである。

（7）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。
（8）並列プロセッサ内に未処理のモジュールが全くなくなった時点で、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だけが並列プロセッサに送り込まれ、並列処理される。
（9）モジュール内の並列処理は、行なわないものとする。
(10) 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、並列化などのオーバヘッドは、無視する。

ここで、仮定（4）は、本論文の目的が並列処理システムの評価でなく、プログラムの並列化手法の評価であることから設定された仮定である。また、仮定（9）はモジュール間の並列処理の効果だけを評価するためにはモジュール間の並列処理を行なうか行なわないかという方式の違いがあればよく、その他の事項については両システムとも同じ処理方式によると仮定すればよいという事実による。

次に、システム解析のために以下の統計的諸量を導入する。

（1）モジュールの実行時間の分布
　モジュールの実行時間の分布を示す確率分布関数 $F_{mod}(t)$ および、確率密度関数 $f_{mod}(t)$

（2）並列度の分布
　並列処理可能なモジュールのグループ内にあるモジュールの数mの確率分布関数 $F_p(m)$ および、確率密度関数 $f_p(m)$

（3）実行ステップ数の分布
　プログラムの実行に当って、並列処理するモジュールのグループの数nの確率分布関数 $F_s(n)$ および確率密度関数 $f_s(n)$

以上のような統計的諸量をもつ並列処理システムにおけるプログラムの実行時間tに関する確率密度関数f(t)を求める。このため、まず、m個の並列処理可能なモジュールからなるモジュールのグループの処理時間tの確率密度関数 $f_c(t)$ を求めると、

$$f_c(t) = a \cdot f_{mod}(t) \cdot (1 - f_{mod}(t))^{m-1} \quad (1)$$

ここでaは、

$$\int_0^\infty f_c(t) \cdot dt = 1$$

となるための正規化係数である。これより、プログラムの処理時間tについての確率密度関数 $f_c(t)$ は、

$$f(t) = \sum_{n=1}^{\infty} \left(f_s(n) \cdot \left(\bigotimes_{i=1}^{n} \left(\sum_{m=1}^{\infty} f_p(m) \cdot f_c(t) \right) \right) \right) \quad (2)$$

となる。$\bigotimes_{i=1}^{n}$ は、n 重のたたみ込みを意味する。

以上のことから、このシステムは、プログラムの到着が平均入のポアッソン到着で、サービス時間の確率密度関数が f(t) であるとして以下でシステム解析する。

4．システム解析

本章では、システムを解析するに当って必要なモジュールに関する統計的諸量を求める。次に、これらの統計的諸量を用いたシステム解析の手法について述べ、解析結果の式を示す。

M/G/1 型待ち行列システムでは、客のサービス時間分布の確率密度関数から、客の平均サービス時間 $1/\mu_0$ と客のサービス時間の2次モーメント b_2 が求まれば、システムのシステム内平均客数L、平均待ち行列長 L_q、客に平均システム内滞在時間Wおよび平均待ち時間 W_q が、以下の式 (3)～(6) によってそれぞれ求められることが知られている。

$$L = \frac{\lambda^2 b_2 + 2\rho - 2\rho^2}{2(1-\rho)} \quad (3)$$

$$L_q = L - \rho = \frac{\lambda^2 b_2}{2(1-\rho)} \quad (4)$$

$$W = \frac{L}{\lambda} = \frac{\lambda^2 b_2 + 2\rho - 2\rho^2}{2(1-\rho)} \quad (5)$$

$$W_q = \frac{L_q}{\lambda} = \frac{b_2}{2(1-\rho)} \quad (6)$$

ここに

$\rho = \lambda/\mu_0$

である。

4．1　モジュールに関する統計的諸量

式(2) から客の平均サービス時間 $1/\mu_0$ と客のサービス時間の2次モーメント b_2 を求めるためには、第3章に記述した三つの統計的諸量（1）モジュールの実行時間の分布、（2）並列度の分布、（3）実行ステップ数の分布、を知る必要がある。

筆者等は、数値計算用サブルーチン IMSL において代入文、DO文、IF文等の各命令の出現頻度が、プログラム中の命令の数に比例してほぼ一定の割合をもっているという調査結果を得ている[9]。IMSLについてモジュール分割を適用し上記の三つの統計データを求めた結果も統計的に安定した性質をもっていた。

（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いて、モジュール分割したときのモジュール、184モジュールにつき実行時間を測定した。図2にモジュールの実行時間の度数分布曲線を示す。また、この確率分布を

$$F_{mod}(t) = 1 - e^{-\mu t} = \int_0^t f(x)_{mod}\, dx \quad (7)$$

のような指数分布とし、実測データから求めた確率分布曲線を図3に示す。これよりパラメータは、

$\mu = 75000$

となる。図中に、この時の確率分布曲線も併せて記述しておいた。

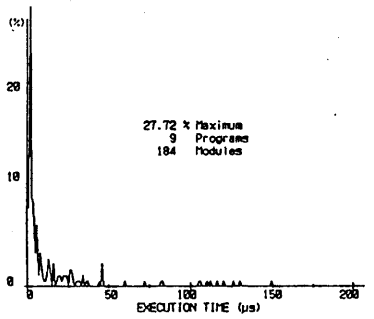

図2　モジュールの実行時間の度数分布

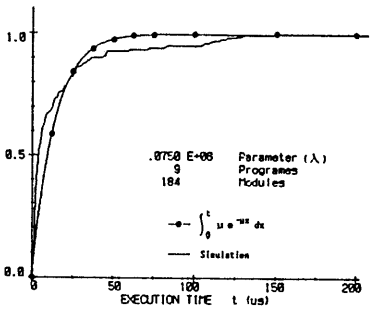

図3　モジュールの実行時間の確率分布

(2) 並列度の分布

361本のプログラムについて、プログラムの開始点から終了点までについて、そこに含まれる 9,792個のモジュールの並列度mの分布を調べた。この度数分布曲線を図4に示す。また、これより並列度mの確率分布を

$$F_p(m) = \sum_{i=1}^{\infty} p_p\,(1-p_p)^{i-1} = \sum_{i=1}^{\infty} f_p(i) \quad (8)$$

のような幾何分布とすると並列度が1である確率は、図5のように、$p_p = 0.771$ である。

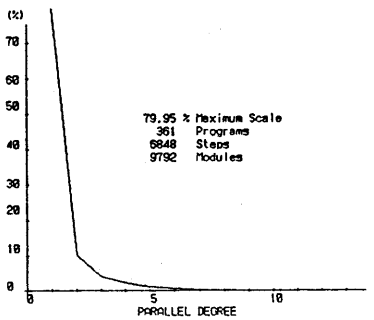

図4　モジュールの並列度の度数分布

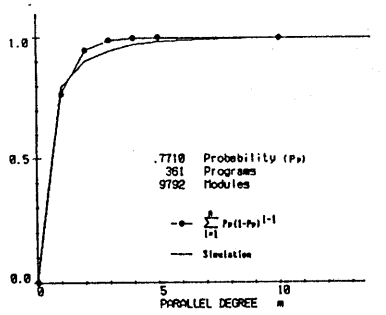

図5　モジュールの並列度の確率分布

(3) 実行ステップ数の分布（並列化した場合）

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一つのパスを選んで、これにモジュール分割手法を適用して並列化する。並列化された後のパスについて、並列処理可能なモジュールのグループの実行を1ステップの実行としてプログラム毎の実行ステップ数を調べる。その結果からステップ数nの度数分布とその確率分布曲線を求めると、図6および図7のようになる。

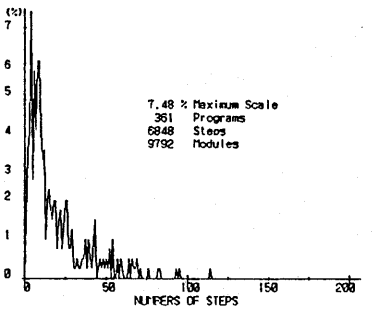

図6　実行ステップ数の度数分布

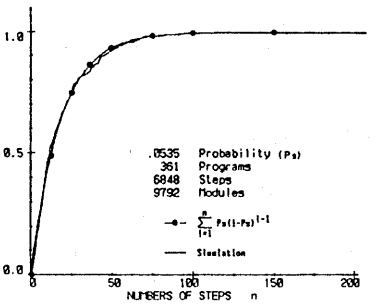

図7　実行ステップ数の確率分布

図のように、分布関数は(9)式の幾何分布において、ステップ数が1である確率は、$p_s = 0.0535$ である。

$$F_s(n) = \sum_{i=1}^{n} p_s (1 - p_s)^{i-1} = \sum_{i=1}^{n} f(i) \qquad (9)$$

（3'）実行ステップ数の分布（並列化しない場合）

（3）と同じプログラム、同じパスについて、並列化しない場合の実行ステップ数を求め、その度数分布と確率分布曲線を得ると、図8および図9となる。

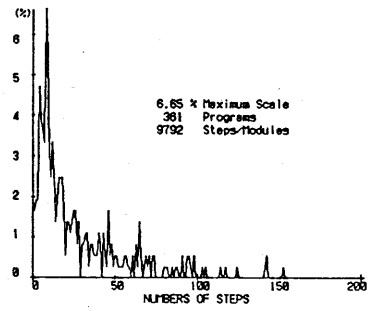

図8　並列化しない場合の実行ステップ数の度数分布

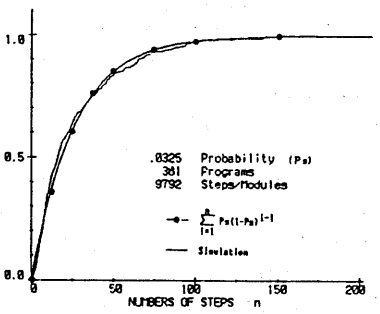

図9　並列化しない場合の実行ステップ数の確率分布

ステップ数nの確率分布関数は、$p_s = 0.0325$ の幾何分布である。

4．2　システム解析手法と解析結果

ここでは、式(2) からその特性関数を求め、モジュール分割を用いた並列化システムにおけるプログラムの平均実行時間 $1/\mu_0$ と、プログラムの実行時間の2次モーメント b_2 を求める。

たがいに独立な確率変数の和の特性関数は、各確率変数の特性関数の積に等しいから、式(2) のたたみ込みの特性関数は、

$$f_1(t) = \sum_{m=1}^{\infty} (f_p(m) \cdot f_c(t)) \qquad (10)$$

の特性関数の積から求まる。

そこで、まず、特性関数

$$\varphi_1(t) = \int_0^{\infty} e^{jtx} f_1(x) \cdot dx \qquad (11)$$

を求めると、

$$\varphi_1(t) = \int_0^{\infty} e^{jtx} \{ \sum_{m=1}^{\infty} p_p (1-p_p)^{m-1} m \cdot e^{-\mu x} (1-e^{-\mu x})^{m-1} \} dx$$

$$= \sum_{m=1}^{\infty} p_p (1-p_p)^{m-1} \mu^m \frac{m!}{\prod_{k=1}^{m} (k\mu - jt)} \qquad (12)$$

となる。従って、式 (2) の特性関数は

$\varphi(t)$

$$= \sum_{n=1}^{\infty} p_s (1-p_s)^{n-1} \{ \sum_{m=1}^{\infty} p_p (1-p_p)^{m-1} \mu^m \frac{m!}{\prod_{k=1}^{m} (k\mu - jt)} \}^n \qquad (13)$$

となる。これより、プログラムの平均実行時間 $1/\mu_0$ とプログラムの実行時間の2次モーメント b_2 を求めると以下のようになる。

$$\frac{1}{\mu_0} = \frac{1}{j} \cdot \frac{d}{dt} \varphi(t) \Big|_{t=0}$$

$$= \sum_{n=1}^{\infty} p_s (1-p_s)^{n-1} \, n \sum_{m=1}^{\infty} p_p (1-p_p)^{m-1} \sum_{k=1}^{m} \frac{1}{k\mu}$$

$$= \frac{-\log p_p}{p_s (1-p_p) \mu} \qquad (14)$$

$$b_2 = \frac{1}{j^2} \cdot \frac{d^2}{dt^2} \varphi(t) \Big|_{t=0}$$

$$= \sum_{n=1}^{\infty} p_s (1-p_s)^{n-1} \, n \{ (n-1) (\sum_{m=1}^{\infty} p_p (1-p_p)^{m-1} \sum_{k=1}^{m} \frac{1}{k\mu})^2$$

$$+ \sum_{m=1}^{\infty} p_p (1-p_p)^{m-1} (\sum_{k=1}^{m} (\frac{1}{k\mu})^2 + (\sum_{k=1}^{m} \frac{1}{k\mu})^2) \}$$

$$= \frac{2(1-p_s)}{\mu_0^2} + \frac{2}{p_s \mu^2} \sum_{n=1}^{\infty} \frac{1}{n} \sum_{m=1}^{n} \frac{(1-p_p)^{m-1}}{m} \qquad (15)$$

並列化しない場合は、$m = 1$、$p_p = 1$ であるから、$1/\mu_0$ と b_2 は、式(14)、(15)から、次式のようになる。

$$\frac{1}{\mu_0} = \frac{1}{j} \cdot \frac{d}{dt} \varphi(t) \Big|_{t=0}$$

$$= \sum_{n=1}^{\infty} p_s (1-p_s)^{n-1} \, n \frac{1}{\mu} = \frac{1}{p_s \mu} \qquad (16)$$

$$b_2 = \frac{1}{j^2}\cdot\frac{d^2}{dt^2}\varphi(t)\Big|_{t=0}$$

$$= \sum_{n=1}^{\infty} p_s(1-p_s)^{n-1}\, n(n+1)\,\frac{1}{\mu^2} = \frac{1}{p_s\mu^2}$$

$$= \frac{1}{\mu_0^2} \qquad (17)$$

以上の解析結果をもとに、システム解析の理論値として、平均システム内滞在時間を求める。このため、４.１の（1）、(2)、(3）に示した統計量μ= 75000、p_p = 0.771、p_s = 0.0535を、式(14)、(15)に代入し $1/\mu_0$、b_2の値を求める。さらにこれを式(6）に代入し、並列化した場合の平均システム内滞在時間Wをプログラムの到着分布に対して求める。

また、４．１の（1）、(3') に示した統計量、すなわち、μ= 75000、p_s = 0.0325 を式(16）、(17)に代入し$1/\mu_0$、b_2 の値を求め、並列化しない場合の平均システム内滞在時間W' を上記と同様にプログラムの到着分布に対して求める。

これらの計算結果については第５章で述べる。

５．システム評価

ここでは、４．１に述べた統計的諸量の実測値を用いて計算機シミュレーションによるシステム解析の結果を並列化した場合と並列化しない場合の両者について求める。次に、第４章で述べた並列化した場合と並列化しない場合のシステム解析の結果から、それらの理論値を求める。そして、シミュレーションの結果に対する理論値の誤差を調べ、理論値の妥当性を評価する。最後に、並列化した場合と並列化しない場合の解析結果を比較してモジュール分割による並列化システムの評価を行なう。

５．１　計算機シミュレーション

計算機シミュレーションは、以下の手順に従って行なう。

（手順１）プログラムのシステムへの到着が、平均入の指数分布に従うとの仮定のもとに、到着時間々隔 t の値をその分布が $1-e^{-\lambda t}$ となるように乱数を用いて求める。

（手順２）サーバ内にモジュールがあるかどうかを調べて、システムに到着したプログラムが、並列プロセッサによって実行可能な状態かどうかを調べる。実行可能ならば（手順３）以下を実行する。他のプログラムが実行中ならば実行待ちのための待ち行列に入れ待機させ、（手順４）により実行中のプログラムの実行を行なう。

（手順３）実行可能なプログラムに関するデータとして361本のプログラムの実測データの中からその一つを、一様乱数を用いてランダムに選ぶ。実測データは、IMSLの中の 361本の任意のプログラムについてそれぞれ任意の一つの実行パスを選び、そのパスについてモジュール分割し、並列化した図10に示すようなデータである。

ﾌﾟﾛｸﾞﾗﾑ番号	ｽﾃｯﾌﾟ数	各ｽﾃｯﾌﾟの並列度
1	12	1 1 2 1 2 2 2 1 3 1 ･･･
2	59	5 1 1 1 1 1 2 5 1 1 ･･･
3	50	7 1 1 1 1 1 1 4 1 2 ･･･
4	9	1 1 2 1 2 1 1 1 1
5	82	3 1 1 1 1 8 1 1 13 1 ･･･
･･･	･･･	･･･････････････････
361	29	3 1 4 4 1 1 2 1 1 1 ･･･

図10　並列度とステップ数の実測データの例

（手順４）システムに到着したプログラムの実行は、手順３によって選ばれたプログラムの実測データにもとずいて、以下に述べるモジュールの実行の繰返しによってなされる。

(a) 実測データによって与えられた並列度の数だけのモジュールの実行時間を、４．１の（1）で与えたパラメータμ= 75000 の指数分布に従うように乱数により設定する。

(b) (a) で与えられた実行時間の内、最大のものをそのステップの実行時間とする。

(c) (a)、(b)をプログラムの各ステップについて実行し、全ステップの実行時間の総和を求めプログラムの実行時間とする。

（手順５）次のプログラムの到着時刻ならば(手順１)へもどり、待機中のプログラムがあれば(手順２)へもどる。そうでないなら次のプログラムの到着まで時間待ちし(手順１)へ。

上記の手順の実行中に、プログラムがシステムに到着してからシステムを離れるまでの時間の平均値、すなわち、平均システム内滞在時間Wを求める。

並列化しない場合の平均システム内滞在時間 W'を求めるには、(手順４）の (b) をそのステップ内にあるモジュールの実行時間の総和を求めるように変えてシミュレーションする。

シミュレーションプログラムは、原理的には上記の手順に従っているが、プログラムの実行時間を短縮するための工夫がなされている。

５．２　並列化システムの評価

まず、WとW' について理論値とシミュレーションの比較結果を表１に示す。シミュレーションの試行回数は約１万である。これらの表からシミュレーションとの誤差が数%以内であり、この解析手法は、十分に実用可能なものである。モジュールの実行時間の測定データ数を

増やせば誤差は、さらに小さくなるものと思われる。

表1　理論値とシミュレーションの結果の比較

到着率	W			W'		
	シミュレーション ($\times 10^{-4}$)	計算値 ($\times 10^{-4}$)	誤差 (%)	シミュレーション ($\times 10^{-4}$)	計算値 ($\times 10^{-4}$)	誤差 (%)
500	3.34	3.30	-1.2	5.23	5.13	-1.9
1000	3.95	3.94	-0.3	7.02	6.86	-2.3
2000	6.59	6.51	-1.2	21.02	22.25	5.9
2300	8.36	8.09	-3.2	66.71	70.50	5.7
3000	18.87	18.70	-0.9	—	—	—
3300	40.54	42.76	5.5	—	—	—

次に、WとW'を比較することによってモジュール分割を用いた並列化の効果を評価する。このため、プログラムの到着率に対してWとW'の理論値を求めると、図11のようになる。これから、処理速度がプログラムの到着率が低いところで約1.5倍、高いところではさらにそれ以上改善されていることが分る。

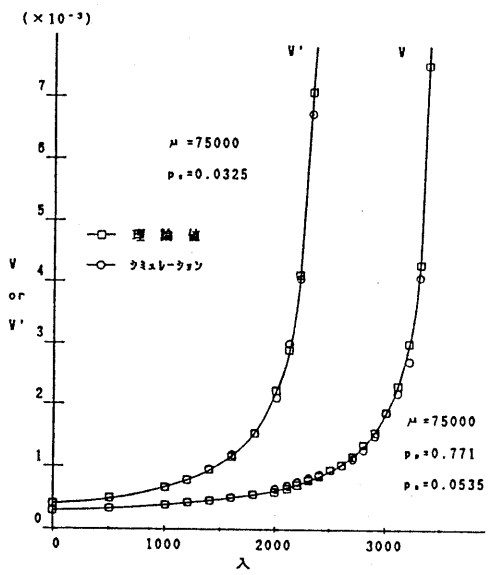

図11　平均システム内滞在時間

6．むすび

本論文では、モジュール分割を用いた並列化システムの評価を目的とし、システム解析手法と並列化の効果について述べた。システム解析手法については、理論値がシミュレーションによる結果と比較して数%程度の誤差であり、解析手法の妥当性が確かめられた。解析結果の式については、一部の項が無限級数で表現されているため、計算機を用いて近似計算しなければならないが、計算の収束速度は、パーソナルコンピュータによっても数秒程度であり計算上の問題はない。しかし、級数を用いない近似式による表現についても検討する必要がある。

並列化の効果については、モジュール分割を用いてプログラムを全体にわたって並列化した場合の並列化の効果が、約2倍以上の処理速度の向上をもたらすことが分った。この並列化手法に加えて、繰返し処理部分に従来の並列化手法を適用すれば、さらに処理効率の向上が図れることになる。また、プログラム中の変数に名前をつける際に、変数の依存関係をできるだけ少なくするなど並列化の効果を上げるためにプログラムを最適化するプログラミング技法を検討する問題がある。

さらに、並列化モジュールキューの長さが有限で、各モジュールバッファは、有限個のモジュールしか受入れられないものとし、並列プロセッサも有限個からなっているシステムについてシステム解析する問題がある。また、並列化手法の検討のためでなく、サーバが、多重プログラム処理可能であるとしてモジュール分割を用いた並列処理システムのシステム解析を行なう問題もある。

参考文献

(1)D.J.Kuck, Y.Muraoka and S.Chen："On the number of operations simultaneously executable in Fortran-like programs and their resulting speedup", IEEE Trans.Comput.,C-21, 12, pp. 1293-1310, (Dec.1972)

(2)D.A.Padua and M.J.Wolfe: "Advanced compiler optimizations for supercomputers", Commun.ACM, 29, 12, pp.1184-1201, (Dec.1986)

(3)安村,梅谷,堀越："自動ベクトルコンパイラにおける部分ベクトル化の方式",情処学論, 24, 1, pp. 15-21 (昭 58-01)

(4)M.Wolfe："Loop skewing : The wavefront method revised", Int.J.Para.Prog., 15, 4, pp. 279-293, (1986)

(5)M.Wolfe and U.Banerjee: "Data dependence and its appication to parallel processing", Int.J. Para.Prog., 16, 2, pp. 137-178, (1987)

(6)C.D.Polychronopoulos, D.J.Kuck and D.A.Padua:"Utilizing multidimensional loop parallelism on large-scale parallel processor system", IEEE Trans. Comput., 38, 9, pp. 1285-1296, (Sept. 1989)

(7)C.McCreary and H.Gill: "Automatic determination of grain size for efficient parallel processing", Commun.ACM, 32, 9, pp. 1073-1078, (May 1989)

(8)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", 信学論(D), J71-D, 8, pp.1525-1531 (昭 63-08)

(9)菊地,白鳥,宮崎："高水準言語によるプログラムの並列化における統計的性質について", 第37回情報処理学会全大 (昭 63-09)

(10)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析について", 第39回情報処理学会全大, (平 1-10)