システムソフトウェアと
オペレーティング・システム 76－5
（1997．8．21）

サービスの処理内容を考慮したトランザクション処理の負荷分散法

青木義則† スカンヤ・スラナワラッ
谷口秀夫 牛島和夫
九州大学 大学院システム情報科学研究科

入出力処理を頻繁に行なうプロセスは、サービスの応答時間を極端に増加させたり、他のサービスへ大きな影響を与える危険性がある。入出力処理の影響を考慮するには、プロセス毎の処理の内容を考慮する必要がある。しかし、プロセスの処理内容を事前に予測するのは難しいため、これまでに入出力処理を考慮した分散法に関する研究はほとんどない。そこで、本稿では、プロセスの特徴を予測することが可能なトランザクション処理を分散対象のサービスとし、分散システムの要素をモデル化し、サービスの応答時間を最小とする分散形態を決定する PS^3(Processing Structure and System Structure) 分散法を提案する。

A Load Distributed Policy with Transaction Service Designed for Minimizing Response Time

YOSHINORI AOKI †, SUKANYA SURANAUWARAT ,
HIDEO TANIGUCHI and KAZUO USHIJIMA
Graduate School of Information Science and Electrical Engineering,
Kyushu University

If the process often accesses the disk, this can possibly increase service response time or have severe effect on other services. To consider the effect due to accessing the disk, we need to know processing content of each process. However, it is hard to know it in advance, that's why there are only a few studies in this field. In this paper, we propose PS^3 (Processing Structure and System Structure) policy that uses a trasaction service of which the content of each process can be estimated in advance. Our objective is to use this policy on a distributed system with various numbers of elements in order to find the minimum service response time.

1. はじめに

近年、高性能な計算機や高速な通信路の普及により、分散システムが一般的になっている。これは、分散システムでは、負荷分散・機能分散による処理効率の向上を図れることが大きな要因である。しかし、この特徴を分散システム上で実現するには、処理の分散法を検討しなくてはならない。

これまでにも多くのプロセスの処理分散法が提案されている[1)~3)]。しかし、処理分散法は、様々な要素を考慮しなくてはならない非常に複雑な問題であるため、これまでの手法では、対象システムを理想的にモデル化し、議論している[1)~3)]。例えば、入出力処理の影響を考慮しないで議論されている。入出力処理の影響を考慮するには、プロセス毎の処理の内容を考慮する必要がある。しかし、プロセスの処理内容を事前に予測するのは困難であるため、これまでの手法の多くでは、プロセスの処理内容が考慮されなかった。しかし、入出力処理の影響を無視して分散形態を決定すると、入出力処理を頻繁に行なうプロセスがボトルネックとなり、サービスの応答時間を極端に増加させたり、他のサービスへ大きな影響を与える危険性がある。

そこで、本稿では、プロセスの特徴を予測することが可能なトランザクション処理を分散対象のサービスとし、分散システムの要素をモデル化する。具体的には、サービス処理を構成するプロセスの処理内容を表現する要素や、計算機の処理能力を表現する要素を明確にする。更に、サービスを構成するプロセスの処理内容に関する情報と、計算機の処理能力に関する情報を基に、プロセスの処理時間の予測の方式について述べる。次に、応答時間を最小とする分散形態を決定する PS^3(Processing Structure and System Structure) 分散法を提案する。PS^3分散法は、複

† 現在: 日本アイ・ビー・エム (株) 東京基礎研究所
now: Tokyo Research Laboratory, IBM Japan, Ltd.

表1 処理構造の要素項目と特徴量

通番	要素項目	特徴量	考慮すべき事項
1	プロセッサ処理	実行命令数：DS	プロセス間通信や入出力との関係
2	プロセス間通信（同期待ち有り）	発生時期と回数、通信データ量	遠隔プロセッサ間か否か
3	プロセス間通信（同期待ち無し）	発生時期と回数、通信データ量	遠隔プロセッサ間か否か
4	入出力（同期待ち有り）	発生時期と回数、入出力データ量	遠隔外部装置へか否か
5	入出力（同期待ち無し）	発生時期と回数、入出力データ量	遠隔外部装置へか否か
6	処理分割規模	プロセス数	（特になし）
7	基本規模	各プロセスの大きさ (bit)	（特になし）

数の協調プロセスによるサービスが複数走行する環境をサポートし、プロセスの処理時間の予測に基づいて応答時間を最小とする分散形態を決定する。また、プロセスを分散するときの方針としては、一度分散し、既に実行がはじまったプロセスを移動しない静的な方式を採用する。

2. 分散システムのモデル化

対象とする分散システムは、通信路で結ばれた複数の計算機のシステムとする。各計算機はそれぞれ一つのプロセッサと一つの外部装置を持っており、サービスを構成するプロセスは、どの計算機でも走行できるものとする。また、各計算機のスケジューリング方式はUNIXなどで採用されているラウンドロビン方式とする。分散の対象とするサービス処理は、複数の協調プロセスで構成されるトランザクション処理とし、プロセス毎の処理内容、処理順序は、事前に把握できているものとする。なお、本稿では、ファイルはローカルファイルに対するアクセスのみを考慮する。

モデル化に際して、分散システムを二つの要素に分ける[4]。一つはサービス処理を実現している部分で処理構造と呼び、もう一つはサービス処理の実行を行なう計算機や通信路の部分でシステム構造と呼ぶ。処理構造の要素項目と特徴量をまとめた結果を表1に示す。システム構造の要素をまとめた結果を表2に示す[4]。

表2 システム構造の要素項目と特徴量

通番	分類	要素項目	特徴量
1		プロセッサ	性能 (MIPS)
2	ハード	外部装置	入出力速度 (bit/sec)
3	ウェア	通信路	通信速度 (bit/sec)
4		分散環境規模	プロセッサ数
5		入出力処理	実行命令数
6	OS処理	通信処理	実行命令数
7		プロセス間通信処理	実行命令数

3. プロセス処理時間の予測

サービスの応答時間は、サービスを構成するプロセスの処理順序および各プロセスの処理時間により決定される。プロセスの処理は、プロセッサ処理、入出力処理、およびプロセス間通信処理の三つである。以降で、プロセッサ処理と入出力処理の処理時間の和をプロセス処理時間と呼ぶ。以降では、協調プロセスのプロセス処理時間の定式化を行なう。

3.1 単一プロセスのプロセス処理時間の予測

プロセスの状態には、大きく分けて、RUN状態、WAIT状態、およびREADY状態の三つがある。プロセスは、RUN状態ではプロセッサ処理を行なう。プロセスは、入出力要求を出すと、入出力要求が既に他のプロセスからあり入出力処理中の場合、プロセスは入出力キューにつながれ、WAIT状態となる。その後入出力処理を行ない、入出力終了後、ハードウェア割り込みによりREADY状態となる。入出力処理中でない場合は、入出力キューでの待ちは発生せず、実入出力によるWAIT状態になる。READY状態となったプロセスは、再びスケジューラによってディスパッチされるまで、待ち状態を続ける。したがって、単一プロセスのプロセス処理時間は、プロセッサ処理時間、入出力キューでの待ち時間、実入出力時間、およびディスパッチ待ち時間の四つの時間の和となる。

3.1.1 プロセッサ処理時間の定式化

計算機 m のプロセッサは、単位時間に処理量 $mips_m$ を処理できるとすると、プロセス i の処理量 $steps_i$ のプロセッサ処理時間 $ptime(m, steps_i)$ は、以下の式(1)で求めることができる。

$$ptime(m, steps_i) = \frac{steps_i}{mips_m} \quad (1)$$

3.1.2 入出力キューでの待ち時間の定式化

入出力キューでの待ち時間は、入出力処理を要求するプロセスの数とプロセス毎の入出力要求の頻度に依存する。ここで、プロセスの入出力処理の頻度を表す指標として、平均入出力間処理量を定義する。プロセス i のプロセッサ処理の総和が $steps_i$ で、入出力処理の回数が $ionum_i$ の場合、平均入出力間処理量 μ_i を以下の式(2)で定義する。

$$\mu_i = \frac{steps_i}{ionum_i} \quad (2)$$

ここで、$qtime$ を入出力キューでの待ち時間、$rtime$ を実入出力時間とするとき、$ptime < rtime$ のプロセスを外部装置依存プロセスと呼び、$ptime > rtime$ のプロセスをプロセッサ依存プロセスと呼ぶ。図1に、外部装置依存プロセスとプロセッサ依存プロセスが共存する場合の入出力処理の様子を示す。図1において、プロセスAはプロセッサ依存プロセス、プロセスB～Dは、外部装置依存プロセスである。プロセスAが入出力要求を時刻 t に出した

とすると、時刻 t に行なわれているプロセスBの実入出力の残りの時間$\Delta rtime$は、$\frac{rtime}{2}$と期待できる。さらに、処理を詳細に見ると、入出力処理要求による入出力キューの処理ipt_1と、入出力終了による割り込み処理ipt_2を行なう。そこで、それらの時間を考慮し、計算機m上で走行する外部装置プロセス数をn_{dk}、外部装置依存プロセスjの計算機mでの実入出力時間を$rtime(m)$とすると、プロセスiの入出力キューでの待ち時間$qtime(m, n_{dk})$は、以下の式(3)で求めることができる。

$$qtime(m, n_{dk}) = (\Delta rtime + ipt_1) + \sum_{j}^{n_{dk}-1} \{rtime(m) + ipt_2\} \quad (3)$$

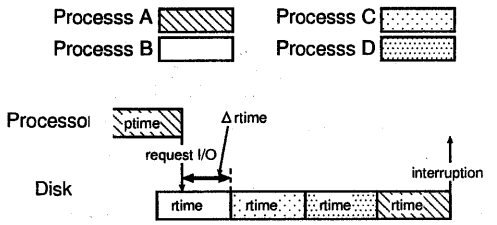

図1　入出力キューでの待ち時間

3.1.3 実入出力時間

実入出力時間は、入出力のデータサイズに依存する。このため、実測による結果を利用する。具体的は実測は1キロバイトのデータの書き込みを行なう処理を繰り返すプロセスを走行させて行なった。このプロセスはμが非常に小さいプロセスである。つまり、外部装置依存プロセスなので、入出力キューでの待ち時間とディスパッチ時間は0と考えられ、この場合の実測時間は、1キロバイトのデータの実入出力時間と入出力終了による割り込み処理ipt_2であるといえる。ここで、ipt_2は非常に小さいので$ipt_2 = 0$とし、実測時間を計算機mでの実入出力時間$rtime(m)$とする。

3.1.4 ディスパッチ待ち時間の定式化

図2を用いて、外部装置依存プロセスとプロセッサ依存プロセスが混在する場合のディスパッチ待ち時間の定式化を説明する。図2において、AとBのブロックは外部装置依存プロセスのプロセッサ利用を示し、C〜Eは、プロセッサ依存プロセスのプロセッサ利用を示している。

ここでは、UNIXのようなOSを想定しているため、入出力終了後のプロセスはプロセス優先度が高い。このため、外部装置依存プロセスは、入出力処理が終了してREADY状態になったらすぐにRUN状態になる。したがって、外部装置依存プロセスのディスパッチ待ち時間は0と考えられる。

一方、プロセッサ依存プロセスのディスパッチ待ち時間は、実際の動き(図2(i))を変換した(図2(ii))もので考える。図2(ii)に示すように、プロセッサ依存プロセスの入出力キューでの待ち時間、実入出力時間、およびディスパッチ待ち時間の和は、他のプロセッサ依存プロセスによる待ち時間$pdpt$と外部装置依存プロセスによる処理時間$ddpt$の和となる。

したがって、ディスパッチ待ち時間$dtime$は以下の式(4)で求めることができる。DKとPUはそれぞれ外部装置依存プロセス、プロセッサ依存プロセスを表現する。

$$dtime(m, \mu_i, n_{cpu}, n_{dk}) = \begin{cases} 0 & ,i=DK \\ pdpt + ddpt - (qtime + rtime), & i=PU \end{cases} \quad (4)$$

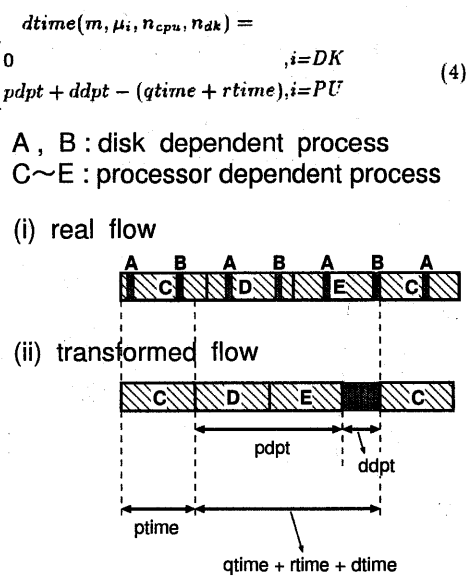

図2　ディスパッチ待ち時間

プロセスiのプロセッサ依存プロセスによる待ち時間$pdpt$は、プロセスiのプロセッサ処理時間$ptime(m, \mu_i)$と、他のプロセッサ依存プロセスの数n_{cpu}に依存し、以下の式(5)で求めることができる。

$$pdpt(\mu_i, n_{cpu}) = ptime(m, \mu_i) \times (n_{cpu} - 1) \quad (5)$$

外部装置プロセスを処理する時間$ddpt$の定式化は、外部装置依存プロセスが単位時間に処理される回数と、外部装置依存プロセスを処理する時間により決定される。実入出力時間を$rtime$とすると、単位時間当たりに外部装置依存プロセスを処理する回数は、$\frac{1}{rtime}$となる。3.1.3項に示した実測では、1キロバイトのデータの書き込みの実入出力時間は1.67ミリ秒であった。そのため、1秒間に処理される実入出力処理の回数は600.3回と予測できる。そこで、外部装置依存プロセスが1秒間に行なえる実入出力処理の回数を実測した。実測結果を図3に示す。図3より、1プロセス当たりの実入出力処理回数は、外部依存プロセス数が増加すると減少するが、平均入出力間処理量の影響はほとんど受けないことがわかる。次に、外部依存プロセス数と外部依存プロセスを1回処理するのに必要な時間の関係を図4に示す。図4より、プロセス数が変化して

も、処理時間は一定であることがわかる。また、平均入出力間処理量が増加すると処理時間も増加するが、この処理時間の増加は、外部装置依存プロセス j のプロセッサ処理時間 $ptime(m,\mu_j)$ である。よって、図3から、$\mu=0$ として求めることができる外部装置依存プロセスの処理時間をθとすると、外部装置依存プロセス jの 1 回の処理時間 $\Delta ddpt(m,\mu_j)$ は、以下の式 (6) で求めることができる。

$$\Delta ddpt(m,\mu_j)=\theta+ptime(m,\mu_j) \qquad (6)$$

また、図3のから求めることができる外部依存プロセス数が n_{dk}の場合の単位時間当たりの処理回数を$\xi(n_{dk})$ とする。図2より、以下の式が成り立つ。

$$ddpt=\begin{array}{l}\sum_j^{n_{dk}}\Delta ddpt(m,\mu_j)\times\xi(n_{dk})\\ \times\{ptime(m,\mu_i)+pdpt(\mu_i,n_{cpu})+ddpt\}\end{array}$$

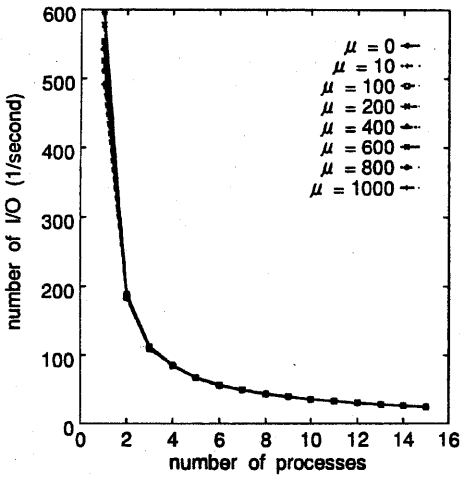

図 3　外部装置依存プロセスの処理回数

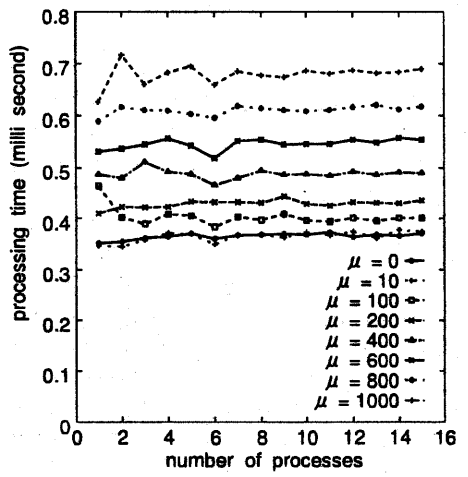

図 4　外部装置依存プロセスの処理時間

3.2　協調プロセスのプロセス処理時間の予測

3.1節により、計算機 m 上で走行する単一プロセス i のプロセス処理時間 $proctime$ は以下の式 (7) で求めることができる。

$$\begin{array}{l}proctime(m,\mu_i,n_{cpu},n_{dk},ionum_i)\\ =\{ptime(m,\mu_i)\\ \quad +qtime(m,n_{dk})\\ \quad +rtime(m)\\ \quad +dtime(m,i,\mu_i,n_{cpu},n_{dk})\}\\ \quad \times ionum_i\end{array} \qquad (7)$$

次に、協調プロセスのプロセス処理時間の予測方式を、図5 を用いて説明する。図5の A～C は、トランザクション・サービスを構成する協調プロセスで、互いにプロセス間通信を行なっている。

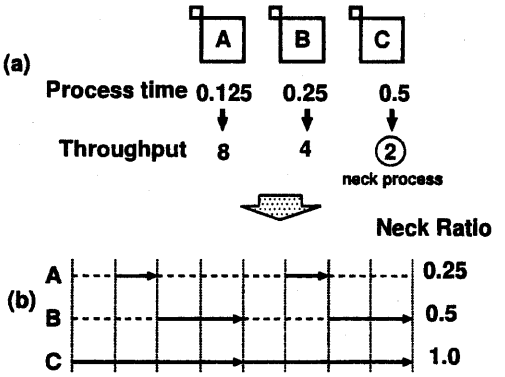

図 5　協調プロセスのネック比

図 5(a) において、プロセス A～C について、単一プロセスとしてのプロセス処理時間を求め、その逆数をスループットと考え、それぞれは 8、4、2 であったとする。このとき、プロセス C がボトルネックとなり、プロセス A と B も、スループットは 2 に制限される。すなわち、サービス全体のスループットは 2 となる。以降では、ボトルネックとなるプロセスをネックプロセスと呼ぶことにする。

プロセスが走行する様子を図 5(b) に示す。矢印はプロセス処理時間を意味し、残りの部分は通信待ち状態を意味する。

ここで、サービス Sのネックプロセス $neck$のスループット th_{neck} から、以下の式でネック比$neckratio_i$を定義する。

$$\begin{array}{l}neckratio_i=\dfrac{th_{neck}}{th_i}\\ \qquad\qquad =\dfrac{proctime_i}{proctime_{neck}}\end{array} \qquad (8)$$

プロセス i のネック比 $neckratio_i$は、任意の時刻 t において、プロセス i が通信待ち状態でない確率を表す。したがって、計算機 m に、任意のある時刻 t において、走行している (通信待ち状態ではない) プロセッサ依存プロセス数の

期待値 n'_{cpu} と外部依存プロセス数の期待値 n'_{dk} は、それぞれ $\sum_i^{n_{cpu}} neckratio_i \times 1$、$\sum_j^{n_{dk}} neckratio_j \times 1$ となる。このため、計算機 m 上で走行する協調プロセス i のプロセス処理時間は、式 (7) において、$n_{cpu} = n'_{cpu}$、$n_{dk} = n'_{dk}$ とすることで定式化できる。

4. PS^3分散法

PS^3 分散法の基本手順を説明する。応答時間が最小となるまで以下の手順を繰り返す。

(1) プロセス決定規則により、移動するプロセスの候補を選ぶ。全てのプロセスが LOCK されていたら、これ以上プロセスを移動できないので終了する。ここで、LOCK とは、計算機への配置が決定したプロセスの状態を意味する。

(2) 計算機決定規則により移動先の計算機の候補を選ぶ。候補となる計算機が無ければ、そのプロセスの移動による応答時間の改善は期待できないと判断し、そのプロセスを LOCK する。

(3) (1) で選んだプロセスを (2) で選んだ計算機に移動した場合のサービスの応答時間とスループットを予測する。

(4) 移動条件を満足していれば、プロセスを移動し、(1) に戻って次の移動の候補となるプロセスを選ぶ。満足していなければ、(2) に戻って次の計算機の候補を選ぶ。

以降では、PS^3 分散法の特徴として、プロセス決定規則、計算機決定規則、移動後の状態の予測、および移動条件について述べる。

4.1 プロセス決定規則

プロセス決定規則では、分散の対象となるサービス処理を構成するプロセスから、移動の対象となるプロセスを決定する。プロセス決定規則を以下に与える。

(1) 未配置プロセスが存在する場合、未配置プロセスの中で最もプロセッサ処理量が大きなプロセスを選ぶ。

(2) 未配置プロセスが存在しない場合、LOCK されていないプロセスの中で、プロセス処理時間が最も大きいプロセスを選ぶ。

4.2 計算機決定規則

移動するプロセスがプロセッサ依存プロセスになるか外部装置依存プロセスになるかは、プロセスの処理構造と移動先計算機のシステム構造に依存する。したがって、このままでは、どちらの優先度を適用して移動先の計算機を選択するか、判断できない。そこで、全ての計算機について、プロセッサ優先度の高い順のリスト (プロセッサ優先度リスト) と、外部装置優先度の高い順のリスト (外部優先度リスト) を作成する。ただし、プロセッサ優先度は、プロセッサ依存プロセスを新たに配置したとき、利用可能なプロセッサ性能を示す指標であり、外部装置優先度は、外部装置依存プロセスを新たに配置したときに利用可能な外部装置性能を示す指標である。これらのリストの先頭を移動先の計算機の候補とする。決定規則の内容は、移動先の候補となる計算機を決定するまで、プロセッサ優先度リストと外部装置優先度リストから交互に計算機を選ぶ。プロセッサ優先度リストと外部装置優先度リストのどちらにも計算機が残っていない場合、移動先の候補となる計算機が存在しないので、プロセスを LOCK する。また、4.4節で述べる移動条件により、選択した計算機へのプロセスの移動が決定した場合、プロセス移動後の状態をもとに、プロセッサ優先度リストと外部装置優先度リストを更新する。

4.3 移動後の状態の予測

ここでは、プロセス移動後について、プロセス処理時間とスループットを予測する方法について述べる。

サービス S を構成するプロセス i を計算機 m_s から m_d に移動した場合について考える。プロセス i を移動することにより、計算機 m_s は走行するプロセス数が減少する。このため、計算機 m_s 上を走行する残りのプロセスは、プロセス処理時間が短くなるものがある。逆に、計算機 m_d ではプロセス i を投入するため、プロセス数が増加する。このため、計算機 m_d 上で走行していたプロセスは、プロセス処理時間が増大するものがある。したがって、計算機 m_s と m_d 上のプロセスのネック比に変化が生じ、サービスのネックプロセスが代わり得る。以上のことから、プロセス移動後の状態を予測するには、全ての計算機について、全プロセスのプロセス処理時間を求めなくてはならない。以下の手順で、全プロセスのプロセス処理時間を求める。

(1) 計算機 m_s と m_d 上の各プロセスのネック比を 1 とし、3.2節で述べたプロセッサ依存および外部装置依存プロセス数の期待値や式 (7)〜(8) により、ネック比の変化がある値 (ここで、値 NR とする) になるまで計算することによって、計算機 m_s と m_d 上の各プロセスのプロセス処理時間を求める。

(2) 計算機 m_s と m_d 上の各プロセスのスループットを求め、さらにそのうちでスループットがもっとも小さいプロセスを見つけ出す。それは、全てのサービスのネックプロセスである。

(3) 式 (8) により、全てのプロセスのネック比を求める。

(4) 計算機 m_s と m_d 以外の計算機上のプロセスについて、3.2節で述べたプロセッサ依存プロセス数および依存プロセス数の期待値や協調プロセスのプロセス処理時間によりプロセス処理時間を求める。

4.4 移動条件

新たに分散するサービスや既存のサービスでは、達成しなければならない応答時間の上限やスループットの下限が存在する場合が多い。これを、それぞれ最大応答時間、最小スループットと名付ける。最大応答時間や最小スループットが設定されている場合の移動条件は以下の通り。

(1) プロセスの移動により、最大応答時間や最小スループットを満たさないサービスが存在する場合、移動

条件は不満足とする。

(2) 全サービスの最大応答時間と最小スループットを満足している場合、プロセスの移動により、分散するサービスの応答時間が小さくなれば移動条件を満足とし、大きくなれば不満足とする。

5. 評　価

ここでは、PS^3分散法で分散形態を決定し、実際にサービス処理を走行させた場合の、プロセス処理時間の予測値と実測値を比較する。

5.1 処理モデル

実測に用いたサービス処理は、トランザクション処理性能評議会が策定するTPC Benchmark ^{TM}Aの標準仕様[5]をもとに、四つのプロセスA～Dで構成するオンライントランザクションの疑似プログラムを作成した。

5.2 実測環境

実測に用いた計算機は、それぞれPentium Pro 180MHz機、Pentium 133MHz機、および486DX2 66MHz機であり、それぞれ固有の外部装置をもっており、10Mbpsのイーサネット型通信路で結ばれている。応答時間の予測において、通信時間は、同一計算機でのプロセス間通信時間を0.2ミリ秒、異なる計算機でのプロセス間通信の時間を1ミリ秒とした。また、4.3節で述べたNRを0.01とした。

5.3 結果と考察

4プロセスからなる一つのサービスを、1サービスから順次増加させ5サービスまで(サービスaからサービスeまで)増やした時について、実測した結果を図6に示す。図6は、サービスaについて同時に走行するサービス数と、サービスaを構成するプロセス処理時間の実測値と予測値の比を、プロセス毎に示している。図6より、プロセスAとDでは、予測値と実測値の差は小さいことがわかる。しかし、プロセスBとプロセスCの予測値と実測値の差には、大きな違いがある。以降に、これらのプロセスBとCの予測値と実測値の差の原因を考察し、明らかにする。

プロセスBはネックプロセスなので、プロセスBのキューにプロセスAから要求された処理が溜る可能性が高い。実測用のプログラムでは、どのプロセスも他プロセスへの処理要求を最大5までとした。したがって、プロセスBの処理時間の実測値は溜った五つの要求された処理の処理時間の和であると考えられる。このため、その実測値を5で割るとプロセスBの予測値と実測値の比がほぼ1に近くなり、プロセスBの予測値と実測値の差は小さいと考えられる。プロセスCは、プロセスDと同様な処理を行ない、実測値の傾向が同じになるべきである。しかし、プロセスCは、プロセスDとは異なり、プロセスBと同計算機上で走行している。このため、プロセスBの影響を受けて、プロセスCの予測値と実測値の差が大きくなっていると考えられる。

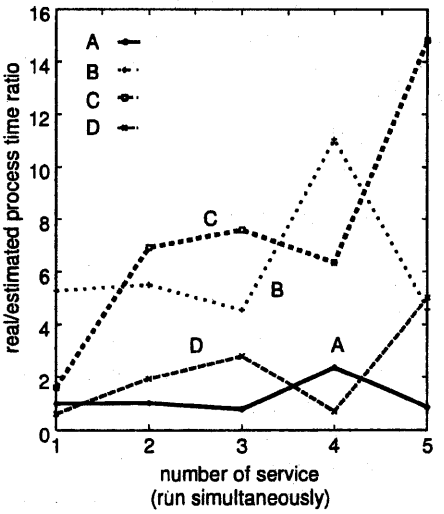

図6 プロセス処理時間の予測値と実測値の比

6. おわりに

本稿では、分散システムを構成する様々な要素を、処理構造とシステム構造に分けて、それぞれの要素項目と特徴量を明確にし、それらを基に、プロセス処理時間を予測する方法を示した。また、プロセス処理時間の予測に基づき、応答時間を最小とする、分散形態を決定するPS^3分散法の詳細を示し、実測と評価を行なった。

今後の課題として、ネック比が小さいプロセスのプロセス処理時間の予測方式の改善、および外部装置からの読み出し処理や通信時間を含めたプロセス処理時間の定式化がある。

参考文献

1) Chow, Y.-C. and Kohler, W. H.: Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System, *IEEE Trans. on Computers*, Vol. c-28, No. 5, pp. 354–361 (1979).
2) Shenker, S. and Weinrib, A.: The Optimal Control of Heterogeneous Queueing Systems: A Paradigm for Load-Sharing and Routing, *IEEE Trans. on Computers*, Vol. 38, No. 12, pp. 1724–1735 (1989).
3) Bonomi, F. and Kumar, A.: Adaptive Optimal Load Balancing in a Nonhomogeneous Multiserver System with a Central Job Scheduler, *IEEE Trans. on Computers*, Vol.39, No.10, pp.1232–1250 (1990).
4) 青木義則, 谷口秀夫, 牛島和夫: 応答時間に着目した静的な処理分散法の実現と評価, 情報処理学会シンポジウム論文集, Vol. 95, No. 7, pp. 117–124 (1995).
5) Gray, J.: The Benchmark Handbook for Database and Transaction Processing System, *Morgan Kaufmann* (1991).