計算機アーキテクチャ 134−17
（1999. 8. 3）

待ち行列網モデルによるSMP型システムの性能予測手法とその精度検証

蔵杉俊康* 海江田章裕† 田中淳裕* 紀一誠*,‡ 堀川隆* 中山泰一†
tk@ccm.cl.nec.co.jp

プロセッサエレメントが安価となるのに伴い、処理速度を向上するためにそれらを複数個搭載したSMP型計算機が普及しつつある。しかしこのSMP型計算機が、サーバマシンなどのように複数プロセスが同時に実行される環境で利用された場合の性能を予測する手法はあまり知られていない。それはSMP型計算機と言うハードウェア、その上で実行されるプロセス、さらにそのプロセスに対する制御という階層的な構造を適切にモデル化し、解析することが難しかったためである。そこで本稿では、待ち行列網モデルを用いてこの階層構造を適切に表現し、その理論解析により性能を予測する手法を提案する。

A Performance Prediction Technique for SMP Computers by Modeling on Queueing Networks

Toshiyasu Kurasugi*, Akihiro Kaieda†, Atuhiro Tanaka*, Issei Kino*,‡,
Takashi Horikawa* and Yasuichi Nakayama†

A technique to predict performance of SMP computers on which multi-processes with critical sections are executed is proposed in this paper. A queueing network proposed in this paper can model processor elements, processes and those critical sections simultaneously. Aanalizing the queueing network theoretically, performance measures of SMP computers can be predicted. We verify the method comparing measured performace in a computer experiment with predicted performance measures.

1 はじめに

プロセッサエレメント(以下 PE)が安価となるのに伴い、処理速度を向上するためにそれらを複数個搭載したSMP型計算機が普及しつつある。サーバマシンなどでは複数のプロセスが同時に実行されることが多く、これらのプロセスを独立かつ並列に実行することができるならばサーバマシンとしてSMP型計算機を採用するメリットは大きい。完全に全てのプロセスが並列に実行できるならば、PE数に対するスループットや速度向上率などの性能は(PE数がプロセス数を越えるまでは)線形に伸びて行くことが予想される。しかし、並列に実行することが許されないクリティカル・セクションがプロセスに存在することも多い。プロセスのクリティカル・セクションはロックやセマフォなどを用いた排他制御により逐次実行される。そのため、クリティカル・セクションの割合が増えると、SMP型計算機のメリットである並列実行性を生かし切れず、期待したほど性能が向上しないことが予想される。

これまでにも、プログラムの処理内容を逐次実行を行う部分と、並列実行可能な部分に分類し、それらの全処理に占める割合からPE数に対する性能を予測する手法が提案されている[1, 2, 5]。しかしこれらの手法では、プログラムの内容として科学計算などを想定しているために、1つのPEが逐次実行部分を処理している時は他のPEはアイドル状態にあると仮定されている。それに対し

* NEC C&C メディア研究所
C&C Media Research Laboratories, NEC Corporation
† 電気通信大学 情報工学科
Department of Computer Science, The University of Electro-Communications
‡ 現在、神奈川大学 情報科学科
presently with Department of Information Science, Kanagawa University

て複数プロセスが同時に実行されるサーバマシンでは、1つのPEがクリティカル・セクションの処理を要求するプロセスを実行している時でも、他のPEはクリティカル・セクション以外の独立かつ並列に実行可能な処理(以下ノンクリティカル・セクション)を要求するプロセスを実行することが可能である。このような、クリティカル・セクションを持つ複数のプロセスが並列にSMP型計算機上で実行されるシステム(以下SMP型マルチプロセスシステム)の性能を適切に予測する手法はあまり知られていない。SMP型マルチプロセスシステムでは、SMP型計算機というハードウェアの上をプロセスが動いており、さらにそのプロセスに対してクリティカル・セクションに伴う制御が行われるいる。つまり、ハードウェア、プロセス、それらのクリティカル・セクションが階層構造を構成しており、性能を精度良く予測するためにはこれらを適切にモデル化する必要がある。そこで本稿では、待ち行列網モデルを用いてこの階層構造を適切に表現し、その理論解析により性能を予測する手法を提案する。

本稿は以下のような構成になっている。まず、2節においてSMP型マルチプロセスシステムを待ち行列網モデルとしてモデル化する。そしてPE数n、プロセス数W、ノンクリティカル・セクション1回当りの平均PE使用時間$1/\mu_1$、クリティカル・セクション 1回当りの平均PE使用時間$1/\mu_2$の4パラメータに対する性能指標の式を示す。次に3節において、実験のためのプログラムを示した上で計算機実験による実測値と提案手法による予測値の比較を行い、最後に4節においてまとめを行う。

2 待ち行列網モデルとその解析

まず、本稿で扱うSMP型マルチプロセスシステムの定義を行うことにする。システムは、同時に実行されるW個のプロセスとn個のPEを搭載する1台のSMP型計算機で構成されているとする。1つのプロセスは、トランザクションを繰り返すとし、トランザクションは1つのクリティカル・セクションと1つのノンクリティカル・セクションで構成されているとする。1プロセスのみをPE数1のSMP型計算機上で実行した場合は、図1の(a)のようになる。処理に要する時間の大半はプロセッサ上で費され、ディスク等他の

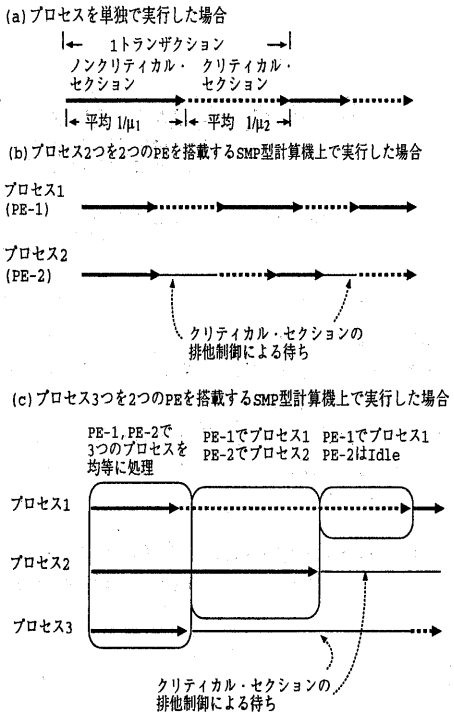

図 1: プロセスの実行状況

資源での処理時間はほとんど無視できるとする。

クリティカル・セクションの処理を同時にPEから受けることができるプロセスの数はシステム上で最大1つと限られており、その実行は先着順(FIFO)で行われるとする。ノンクリティカル・セクションの処理を同時に受けられるプロセスの数には制限がない(つまりプロセスは完全に独立にかつ並列に実行される)とする。例えば、図1の(a)で示されたプロセス2つを2つのPEを搭載したSMP型計算機で実行した場合は、各プロセスは図1の(b)のようにPEから処理を受けることになる。また図1の(c)のように、(クリティカル・セクションに対する排他制御により実行待ちにあるプロセスを除いた)プロセスの数がPEの数よりも大きい場合は、プロセッサシェアリング(以下PS)の規律によりそれらのプロセスは均等にサービスを受けるとし、PE数よりも実行中のプロセスの数が少ない場合は、実行中の各プロセスは1つのPEを専有して処理を受け、余っているPEはIdle状態にあるとする。またシステムのスループットとは、単位時間当りに処理されたト

ランザクションの数の平均を指すものとする。

このようなSMP型マルチプロセスシステムの例としては、Linux 2.0を挙げることができる。Linux 2.0ではシステム・コールを実行する際に、カーネル全体を1つの共有資源と考え、ロックによる排他制御を行っている。カーネル・モードをクリティカル・セクションと、ユーザ・モードをノンクリティカル・セクションと置き換えれば、Linux 2.0は、上記で定義したSMP型マルチプロセスシステムとして捉えることができる[4]。

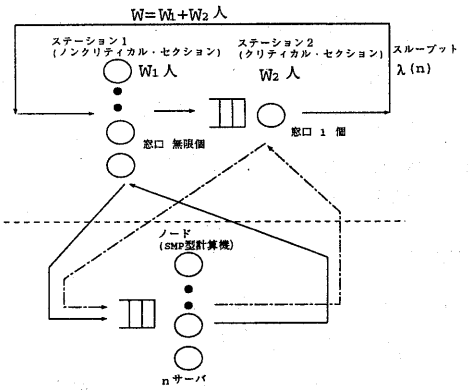

図 2: 待ち行列網モデル

ここで定義したSMP型マルチプロセスシステムは以下のような待ち行列網モデルとしてモデル化することができる(図2)。

- 2つのステーション、1つのノードとそれらを循環するW人の客から構成される。
- ステーション1は無限個の窓口で構成され、ステーション2は1個の窓口と1つのバッファで構成される。
- ステーションに到着した客は、空いている窓口からノードに進む。その窓口はその客がノードから戻るまで占有される。窓口が全て占有されている場合、客はバッファに並んで待つ。
- ノードには、n個のサーバがあり、客はそれらのサーバにおいてPSの規律でサービスを受ける。
- ステーション1からの客はノードに、平均$1/\mu_1$の時間のサービスを要求し、ステーション2からの客はノードに、平均$1/\mu_2$の時間のサービスを要求する。
- ノードでサービスを受け終わった客はもとの窓口へ戻る。
- 窓口に戻った客は、窓口の占有をやめ、もう一方のステーションへ進む。

2層型待ち行列網モデルでの状態は、ステーション1にいる客数(=ノードにいるステーション1からの客数)W_1、ステーション2にいる客数W_2、およびノードにいるステーション2からの客数$L_2 = \min(1, W_2)$により記述することができる。ここで、$W_2 = W - W_1$, $L_2 = \min(1, W - W_1)$の関係が成り立つので、実際にはモデルの状態はW_1のみで表現することができる。ここでさらに、客がノードにおいて要求するサービス時間が指数分布に従うと仮定する。すると、モデルの状態がW_1としてあたえられた場合、ステーション1における客の平均スループット、ステーション2における客の平均スループットはそれぞれ、

$$\tau_1(n, W_1) = \frac{W_1}{\max(n, W_1 + L_2)} n\mu_1 \quad (1)$$

$$\tau_2(n, W_1) = \frac{L_2}{\max(n, W_1 + L_2)} n\mu_2 \quad (2)$$

となる。W_1の状態推移は、図3のような出生死滅過程になり、ノードのサーバ数nのモデルでの状態W_1の定常分布は、

$$\begin{aligned} \Pr(W_1 = 0) &= C(n) \\ \Pr(W_1 = w_1) &= C(n) \prod_{k=1}^{w_1} \frac{\tau_2(n, k-1)}{\tau_1(n, k)} \\ C(n) &= 1 \Big/ \left(1 + \sum_{w_1=1}^{W} \Pr(W_1 = w_1)\right) \end{aligned} \quad (3)$$

と求まり、ステーション1またはステーション2での客のスループット$\lambda(n)$は、

$$\begin{aligned} \lambda(n) &= \sum_{w_1=0}^{W} \Pr(W_1 = w_1)\tau_1(n, w_1) \\ &= C(n)\mu_2 \sum_{w_1=0}^{W} \prod_{k=1}^{w_1-1} \frac{\tau_2(n, k)}{\tau_1(n, k)} \end{aligned} \quad (4)$$

となる。また、並列計算機の評価において用いられることが多い速度向上率および効率も

$$S(n) = \lambda(n)/\lambda(1) \quad (5)$$
$$E(n) = S(n)/n \quad (6)$$

により算出することができる。

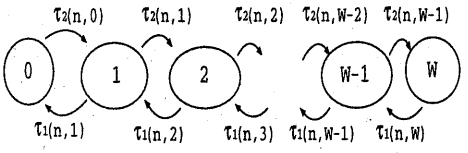

図 3: 状態 W_1 の推移図

3 計算機実験による検証

3.1 プログラム

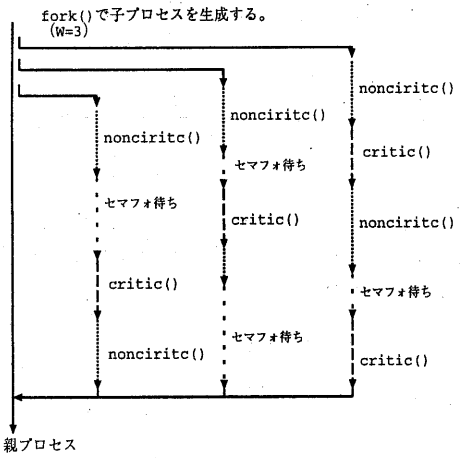

図 4: 実験プログラム

予測手法が実際の SMP 型計算機上での性能予測に有効かを検証するために以下のようなセマフォを用いた排他制御を行うプログラムを作成した (図 4)。

1. fork システムコールにより W 個の子プロセスが生成される。親プロセスは、指定される測定時間だけ待ち、子プロセスを終了させる。
2. 子プロセスは noncritic 関数と critic 関数を繰り返す。
3. noncritic 関数は独立かつ並列に実行される。
4. critic 関数の実行は、セマフォを用いて FIFO ルールで排他制御される。
5. 終了した子プロセス i $(= 1 \sim W)$ はそれぞれ、noncritic 関数の実行に要した時間の合計 NC_i と (セマフォ獲得までの待ちを除く) critic 関数の実行に要した時間の合計 C_i と、セマフォ獲得までの待ち時間の合計 L_i と、critic 関数を処理した回数 Th_i を出力する。

ここで、noncritic 関数は、ノン・クリティカルセクションに相当し、その実行毎に、

step N-1 平均 R_1 となる指数乱数を生成し、この値を r_1 に代入、

step N-2 r_1 個の乱数を生成

を行う。また、critic 関数は、クリティカル・セクションに相当し、その実行毎に

step C-1 セマフォを要求する。獲得するまで実行可能状態で待つ。

step C-2 平均 R_2 となる指数乱数を生成し、この値を r_2 に代入する。

step C-3 r_2 個の乱数を生成する。

を行う。noncritic 関数の **step N-2** と critic 関数の **step C-3** で乱数を生成するのは、クリティカル・セクション、ノンクリティカル・セクションで適当な負荷を PE に与えるためであり、PE に負荷を与える処理であるならば他の処理に置き換えて問題無い。また、noncritic 関数の **step N-1** と critic 関数の **step C-2** で指数乱数を生成するのは、それぞれの実行時間の平均が指数分布にしたがうようにするためである。乱数の生成 1 回当りに要する平均時間を c とすると、$1/\mu_1 = cR_1$, $1/\mu_2 = cR_2$ の関係が成立つので、R_1, R_2 を調整することで、μ_1, μ_2 を調整できる。

3.2 実験による検証

上記プログラムを SMP 型計算機上で実行した場合の実測値と、本稿で提案した手法による予測

値の比較を速度向上率を用いて行った。SMP 型計算機は、SuperSparc+(60MHz) を 8 つ搭載した SPARC server 1000E を使用した。1MB の 2 次キャッシュメモリ、512MB の主記憶を持っており、OS は Solaris 2.5.1 である。

クリティカル・セクションの占める割合に対応して予測精度が異る可能性があると考え、その割合を 4 通りに変えながら実験を行った。具体的にはノンクリティカル・セクションでの乱数発生の平均個数を $R_1 = 100000$ と固定し、クリティカル・セクションでの平均個数を $R_2 = 10000, 20000, 40000, 60000$ とした 4 ケースについて PE 数 n に対するスループットの実測し、それらの値からの速度向上率の算出を行った。全処理に占めるクリティカル・セクションの割合を

$$\rho = R_2/(R_1 + R2) \left(= \frac{1/\mu_2}{1/\mu_1 + 1/\mu_2}\right)$$

で表すことにすると、それぞれのケースは、$\rho = 1/11, 1/6, 2/7, 3/8$ となる。

(実測)

計算機の稼働 PE 数 1 ～ 8 に対するスループットの実測を各ケースに対して $W = 16$ でプログラムを実行して行う。稼働 PE 数が n の時のスループット $\sum_{i=1}^{W} Th_i$ を $\lambda(n)$ とし、式 (5) を用いて速度向上率を算出する。各ケースに対する結果が図 5～ 8に示されている。1 ケースに対して 6 回の測定を行い、速度向上率の平均値と 95% 信頼区間の上界と下界をプロットしている。ここで、測定時間は 180 秒としている。

(予測)

計算機の稼働 PE 数を $n = 1$ にし、$W = 1$ でプログラムを実行する。$\mu_1 = Th_1/NC_1, \mu_2 = Th_1/C_1$ とし、μ_1, μ_2 を得る。また、この時の L_1/Th_1 は、$1/\mu_1$, $1/\mu_2$ に比べると桁が 2 桁以上小さくなり、(待ちを除いた) セマフォの制御に要する時間はほぼ無視できることがわかる。

このようにして求めた μ_1, μ_2 と $W = 16$ を式 (1) ～ (5) に代入し、PE 数 n に対する速度向上率の予測を行う。各ケースに対して 6 回ずつ予測を行い、その平均値が予測平均として図 5～ 8にプロットされている。

(比較)

各ケースに対する平均予測値と平均実測値の相

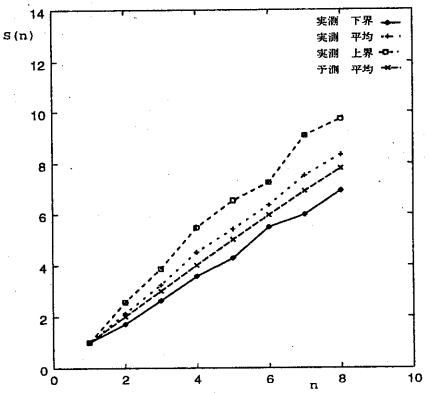

図 5: $\rho = 1/11$

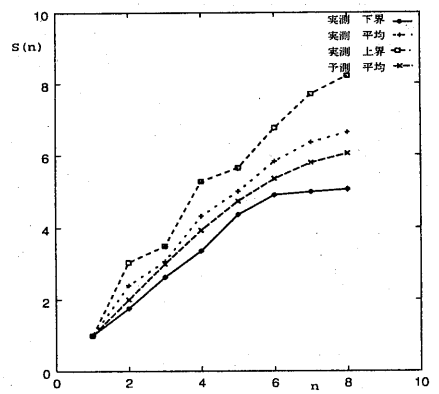

図 6: $\rho = 1/6$

対誤差

(予測値 − 実測値)/実測値 × 100(%)

を求めると表 1に示されるようになる。表および図 5～ 8から、予測値と実測値の平均は近く、予測値の平均は実測の 95% 信頼区間に収まるため、複数回予測を行うことで適切な誤差内で (実測値の平均の) 予測が可能であるとわかる。ρ による精度の違いは特に見られなかった。

4 まとめ

待ち行列網を用いてモデル化することで、SMP 型計算機、プロセス、プロセスのクリティカル・セクションに伴う排他制御を適切に表現すること

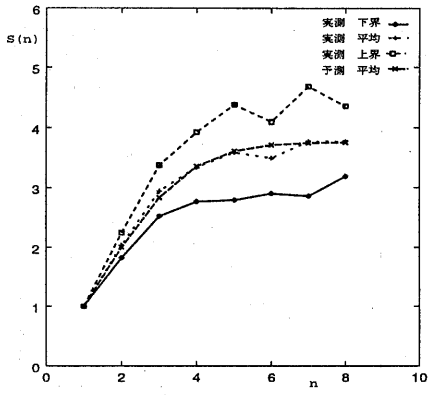

図 7: $\rho = 2/7$

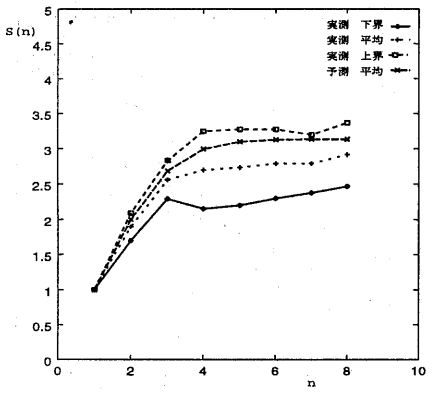

図 8: $\rho = 3/8$

が可能となった。さらにこのモデルを解析することにより、SMP 型マルチプロセスシステムの性能を適切な誤差内で予測できることが確認できた。Linux 2.0 などを OS とした場合の SMP 型計算機の性能予測などにこの手法を適用してみたいと考えている。

実際のアプリケーション・プログラムを用いた実測からも、クリティカル・セクションが SMP 型計算機の性能に大きく影響を与えることが示されている [3]。ここで挙がられているアプリケーションの性能予測にも本手法を適用してみたいが、モデル化で考慮に入れていない I/O 処理がこれらのアプリケーションには存在するために、このままでは適用することができない。I/O 処理に対する拡張を加え、予測対象とするシステムを広げて行きたい。

表 1: 相対誤差 (%)

	R_1			
PE 数	10000	20000	40000	60000
1	0.00	0.00	0.00	0.00
2	−6.06	−16.30	−1.63	5.70
3	−7.38	−2.32	−3.83	4.82
4	−11.37	−9.20	0.43	10.90
5	−7.70	−5.57	0.71	13.26
6	−6.18	−8.15	6.18	12.11
7	−8.13	−8.83	−0.68	12.44
8	−6.35	−8.91	−0.45	7.45

また、プログラムの動作はモデルと非常に近いにも関わらず、予測精度にはぶれがあり、正確とまでは言えない。これは、実際のコンピュータシステムにおいては今回のモデル化では表現できていないことが起こっているためと思われる。さらに適切なコンピュータシステムのモデル化について今後も検討して行きたい。

参考文献

[1] G. Amdahl: Validity of the single-processor approach to achieving large scale computing capabilities, *Proc. AFIPS Conf.*, pp.483–485 (1967).

[2] J. Gustafson: Reevaluating Amdahl's Law, *CACM*, Vol.31, pp.532–533 (1988).

[3] 海江田章裕, 中山泰一, 田中淳裕, 堀川隆, 蔵杉俊康, 紀一誠: ロックの比率に着目した並列プログラムの分類, 情報処理学会 HPC 研究会報告, 99-HPC-77-26 (1999).

[4] 蔵杉俊康, 紀一誠: UNIX を OS とする並列計算機システムの性能予測手法, 情報通信ネットワークの新しい性能評価法シンポジウム, pp.21–30 (1999).

[5] 古市実裕, 永松礼夫, 出口光一郎: 高並列計算機の性能評価のための挙動予測モデル, 情報処理学会論文誌, Vol.38, No.9, pp.1736–1744 (1997).