ハイパフォーマンス
コンピューティング 77－25
（1999．8． 3）

ロックの比率に着目した並列プログラムの分類

海江田 章裕† 中山 泰一† 田中 淳裕†† 堀川 隆†† 蔵杉 俊康†† 紀 一誠††,☆
kaieda-a@igo.cs.uec.ac.jp

SMP 型計算機が普及し、それに対応した UNIX が広く使われるようになってきた。このような並列処理環境で、複数のプロセスからなる並列プログラムを動作させる機会は増えてきている。本論文では、このような並列プログラムの分類についてロックの比率に着目した方法を提案する。ロックの比率とは、あるプロセスの全実行時間に対するクリティカル・セクションの実行時間の割合である。並列プログラムでは、このロックの比率が高いことが性能低下の大きな原因になると考えられる。特に Linux 2.0 では、カーネルを共有資源と考えて、それ全体にロックをかけているため、システム・コールを呼び出すことが、ロックをかけることを意味する。この OS 上でロックの比率の異なる 3 種類の並列プログラム、行列乗算、make の並列実行、WWW サーバを動作させ、ロックの比率と性能向上の関係を調べた。その結果、ロックの比率が高くなると、性能が低下することが確認された。本論文で提案する分類は有効であることが示された。

Classification of Parallel Programs by Analyzing Lock Ratios

Akihiro Kaieda,† Yasuichi Nakayama,† Atsuhiro Tanaka,††
Takashi Horikawa,†† Toshiyasu Kurasugi†† and Issei Kino††,☆

This paper presents how to classify parallel programs by analyzing lock ratios. A lock ratio for a program is defined as a ratio of critical sections to a total execution time of a process. The lock ratio determines how effective programs work on SMP (Symmetric MultiProcessor) systems. We have measured performance of three types of parallel programs: matrix multiplication, parallel make and WWW server, all of which have different lock ratios. Experimental results show that the higher lock ratio parallel programs have, the worse their performance become.

1 はじめに

対称型の密結合並列計算機である SMP 型計算機がパーソナル・コンピュータ (PC) の分野にも普及してきた☆☆。オープン・ソースな UNIX、たとえば、Linux や FreeBSD は、SMP 型計算機で UNIX プロセスを並列実行できるように拡張されてきている。

複数のプロセスからなる並列プログラムは、サーバなどの用途として数多くみることができる。一般的に、このような並列プログラムを SMP 型計算機で動作させれば、それらプロセスが独立かつ並列に実行され、より良い性能を得られると考えられている。しかし、すべてのプロセスが完全に並列実行されていれば良いが、実際には、各プロセスが不可分な共有資源を利用しようとする場合、排他制御を行い、逐次実行されるので性能が低下してしまう。そのため、ある並列プログラムをプロセス・レベルの並列性で考えると、非常に並列度が高いようにみえても、実は、それほど、並列度が高くないということがありえる。

従来、並列プログラムの性能を評価する時は、そのプログラムの逐次実行部分と並列実行部分を分けて、性能を評価していた [1, 3, 4, 9]。しかし、大部分が科学技術計算を前提としており、システムの共有資源を利用することや、それに伴うロックについては想定していない。

† 電気通信大学 情報工学科
Department of Computer Science, The University of Electro-Communications
†† NEC C&C メディア研究所
C&C Media Research Laboratories, NEC Corporation
☆ 現在、神奈川大学 情報科学科
Current affiliation: Department of Information Science, Kanagawa University
☆☆ Intel 社が公開した仕様 (MultiProcessor Specification Version 1.4 July 1995) による。

そこで、本論文では、その指標としてロックの比率 (ロック率) に着目し、それを用いて、並列プログラムを分類することを提案する。ロックの比率とは、あるプロセスの実行時間に対する、ロックを獲得してからロックを解放するまでの排他制御された部分の実行時間の割合である。

ロック率により並列プログラムが分類されていれば、並列プログラムの設計者は、それを利用しておよそどれぐらい性能向上するか見積もることができる。

特に最近、よく用いられる Linux 2.0 などでは、カーネル全体を不可分な共有資源としており、プロセスが、システム・コールの呼び出しでユーザ・モードからカーネル・モードへ遷移するとき、カーネル全体にロックをかけて処理が行われる。この特徴から、Linux 2.0 のロック率はシステム・コールの実行時間の比率と考えることができる。あるプロセスがシステム・コールを多く呼び出すと、並列プログラムの性能が大きく低下すると考えられる。

Linux 2.0 で動作するロック率の異なる 3 種類の並列プログラム、行列乗算、make の並列実行、WWW サーバについて、ロックの比率と性能向上との関係を調べた。その結果、ロックの比率が高くなると、SMP 計算機では、性能が低下することが確認された。本論文で提案するロックの比率に着目した分類が有効であることが示された。

2 関連研究

従来、並列プログラムの性能を評価する場合は、そのプログラムの設計者によって明示的に作られた逐次実行部分と並列実行部分から性能向上率を計算することが多かった。この方法は科学技術計算の性能評価に用いられることが多い [1, 3, 4, 9]。しかし、応用プログラムの特徴とベクトル型計算機であるかなどのハードウェアの構成でのみ評価しており、応用プログラムがどれぐらい多くシステムの共有資源 (たとえば I/O 処理) を利用しているかは考慮していない。

オペレーティング・システム (OS) の性能評価として、システム・コールの実行時間、プロセスのコンテキスト・スイッチにかかる時間、ファイル・システムやネットワークの性能を Solaris、FreeBSD、Linux、Windows など PC で動作する OS でそれぞれ性能を比較している研究がある [2, 7]。これらの値から、あるプログラムを動作させた場合の傾向を知ることができる。しかし、そのような研究は単一 CPU のシステムの結果であるため、SMP 型計算機では、どのような性能を示すかどうかについては分からない。また、OS のみの評価が中心であり、応用プログラムとの関係を考慮してない。

また、我々は SMP 型計算機上で並列プログラムを実行した場合の性能予測手法を待ち行列網モデルを用いて行うことを提案している [6]。プロセスを客に、ロック管理を行う領域を上位ステーションに、ロックをステーションの窓口に、共有資源を下位層のノードとして、OS を待ち行列網モデルでモデル化した。このモデルを用いて、排他制御のある OS を用いた並列プログラムの性能を精度良く予測することができることをシミュレーションで確認している。この手法は、SMP 型計算機というハードウェア上で、応用プログラムと OS をロックによって結びつけて性能評価することを目指している。

本論文では、ロック率に着目すれば並列プログラムの分類ができることを、実際の並列計算機、OS、応用プログラムを用いて実験的に確認する。その結果から、並列プログラムの性能向上をロック率で判断できることを示す。

3 ロックの比率に着目した並列プログラムの分類方法の提案

SMP 型計算機に対応した OS では、システムの共有資源は各プロセスから同時に利用される可能性があるので、カーネル内で排他制御をして管理することが多い。この排他制御された部分をクリティカル・セクションと呼ぶ。

プロセスがその共有資源を利用する通常の動作を説明する。まず、システム・コールなどを呼び出してユーザ・モードからカーネル・モードに状態を遷移する。そして、カーネル・モー

表 1: ロック率と性能向上率の関係の予測

ロック率	低い	中程度	高い
並列プログラムの例	行列乗算	make の並列実行	WWW サーバ
特徴	ユーザ・モードで計算することが大部分	ソース・ファイルを読み込む、オブジェクト・ファイルを書く	HTML ファイルを読み込む、ネットワークを利用
性能向上の予測	高い	中程度	低い

ドの操作で、共有資源を利用するためのロックを獲得して、クリティカル・セクションに入り処理を行う。処理が終了すればロックを解放して、再びユーザ・モードに遷移する。

ロック率とは、あるプロセスの全実行時間に対する、ロックを獲得してから解放するまでのクリティカル・セクションで行う処理の時間の割合のことである。本論文では、そのロック率に着目して並列プログラムを分類することを提案する。単一 CPU の計算機で、ある並列プログラムのロック率について詳細な測定・解析を行い、そのプログラムが CPU 数に見合った性能向上を得られるか判断することができる。

特に Linux 2.0 などでは、カーネルを不可分な共有資源とし、それがクリティカル・セクションとなっている。そこで、あるプロセスが、たとえばシステム・コールの呼び出しでユーザ・モードからカーネル・モードへ遷移するとき、カーネル全体にロックを掛けて処理を行われる (図 1)。ロックの比率はほぼシステム・コールの比率と考えられる。それ以外で、カーネル・モードで行う処理は、割り込み処理、特にプロセス・スケジューリングが挙げられる。

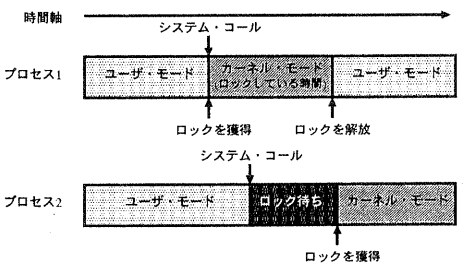

図 1: Linux 2.0 のカーネル・ロック

このため複数のプロセスがほぼ同時にシステム・コールを呼び出した場合、一つは、ロックを掛けてカーネル・モードに遷移できるが、他は、それの処理が終るまで待つことになる。したがって、このロックが多いような並列プログラムでは、性能低下の大きな原因になりうると考えられる。

並列プログラムのロック率による分類がされていれば、並列プログラムの設計者は、システム・コールの呼び出す割合から、およそどれぐらい性能向上するか見積もることができる。

ロック率は、プログラム設計者が明示的にシステム・コールを呼んでいない場合でも、高くなることが考えられる。ユーザ・レベルで記述されたライブラリであっても内部でシステム・コールを呼び出していることがある [8]。

表 1 は、ロック率に着目し、並列プログラムを分類したものである。ロック率の高さで分類し、その代表的な並列プログラムを挙げて、その性質から性能向上を推定した。

ロック率が低い並列プログラムとは、ユーザ・モードで実行をすることが大部分のもので、システム・コールは少ない。そのため、性能向上率は非常に高いと思われる。代表例として行列乗算プログラムが挙げられる。また、ロック率が中程度の並列プログラムとは、ファイル操作などに多少のシステム・コールを呼び出すものであり、ロック率の低い並列プログラムほどの性能向上は得られないと予測される。代表例として make の並列実行が挙げられる。最後にロック率の高い並列プログラムであるが、これは、それぞれのプロセスがシステム・コールを呼び出して、ディスク、ネットワークなど共有資源を利用することが非常に多く、その結果、ロック率

表 2: ロック率と速度向上 (速度向上 = 1 CPU の実行時間/ 2 CPU の実行時間)

並列プログラム	行列乗算	make の並列実行	WWW サーバ
ロック率	0.1%未満	5.11%	51.7%
速度向上	1.95 倍	1.82 倍	0.91 倍

が高くなってしまうものである。これは、SMP 型計算機で動作させても、ほとんど性能向上を得られないと予測される。代表例として WWW サーバが挙げられる。上記の分類が有効であることを、実験により検証する。

4 実験

4.1 実験対象

ロック率が異なると考えられる並列プログラム、行列乗算プログラム、make の並列実行、WWW サーバの 3 種類について実験した。これらの並列プログラムは複数のプロセスが独立して処理をするようになっており、各プロセスで並列性がある。この 3 種類の並列プログラムについて、ロック率を測定し、実行時間から計算される速度向上との関係を調べ分類をする。

実験に用いた計算機は SMP 型の PC で CPU は PentiumPro 180MHz を 2 台持ち、それぞれの CPU は、256KB の 2 次キャッシュを内蔵している。メモリは 128MB 搭載している。また、OS には Linux 2.0.36 を用いた。

Linux 2.0 を用いると、システム・コールの実行時間の割合をロック率とみることができるため、ロック率を把握しやすい。並列プログラムの設計者は、システム・コールの割合から、そのプログラムがどの分類に当てはまりそうかが、容易に予測できる。また、Linux はオープン・ソースなので、ロック率計測用のコードをカーネル内に容易に埋め込むことができる。

4.2 システムの挙動の詳細な測定方法

ロック率は、システムの挙動から計算する。システムの挙動とはシステム・コールの実行などのプロセスのイベントである。これは、カーネル内に挿入したソフトウェア・プローブの情報 (イベントの種類とそのイベントの開始時間) を記録して測定する。

ソフトウェア・プローブのみの計測は、データの記録の仕方などでオーバヘッドが大きくなるので、カーネルのイベントを詳細に計測する場合は利用できない。ソフトウェアのみの計測には、ユーザ・モードでの計測にシステム・コールを使う場合などが挙げられる。

また、SMP 型計算機で計測をする場合、CPU 内のローカルなカウンタを利用することはできない。ローカルなカウンタは、CPU 間で値を一致させて、同じ時間軸で管理することができないからである。

これらの問題を解決するため、ソフトウェア・プローブとハードウェア・トレーサのハイブリット・モニタ手法を用いてロック率を測定する [5]。この手法はソース・コードにソフトウェア・プローブを埋め込み、ソフトウェア・イベントを検出する。その測定データは、バスの信号でハードウェア・トレーサに送られる。そして、ハードウェア・トレーサは測定データを記録する。ソフトウェア・プローブのほうは、メモリ・マップされた特定ポートに書き込むだけなので、ソフトウエア・プローブのオーバヘッドを低くすることができる。また、SMP 型計算機のような場合、イベントの発生時刻を同じ計測用の時計で管理しなければならないが、この手法であれば、ハードウェア・トレーサが持つ時計で時間を管理できる。

4.3 実験結果

3 種類の並列プログラムのロック率とその速度向上を計測した (**表 2**)。以下、各並列プログラムの実験結果を述べる。

4.3.1 行列乗算プログラム

1000 × 1000 の行列乗算プログラムを実行した。行列計算は部分行列の独立したブロックとして計算できる。その部分行列をプロセスに分

表 3: 行列乗算のロック率と速度向上

1 CPU	2 CPU	速度向上	ロック率	ロック待ち率
415(秒)	213(秒)	1.95 倍	0.1%未満	0.1%未満

表 4: make の並列実行のロック率と速度向上

1 CPU	2 CPU	速度向上	ロック率	ロック待ち率
618(秒)	340(秒)	1.82 倍	5.11%	1.61%

表 5: WWW サーバのロック率と速度向上
(64 クライアント)

サイズ	1 CPU	2 CPU	速度向上	ロック率	ロック待ち率
15KB	0.33(秒)	0.34(秒)	0.97 倍	28.1%	22.6%
150KB	1.34(秒)	1.47(秒)	0.91 倍	51.7%	30.4%

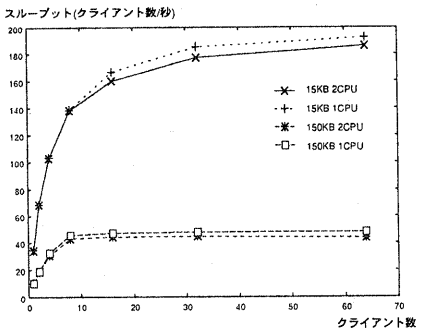

図 2: Apache 1.3.6 のスループット

割して計算させるので、プロセスは独立、かつ、並列に動作できる。実験では、CPU 数に合わせて、行列を 2 つに分割して、それを 2 つのプロセスが計算するようにした。

計算結果の領域は、どのプロセスからもアクセスできるように、mmap システム・コールを用いた共有メモリ空間に置いた。この領域へのアクセスは、計算結果が独立しているので排他制御が必要ない。

ロック率は 0.1%未満でとても低い。これは、プロセス生成、共有メモリ空間の獲得など、初期化処理のみで、計算実行中はシステム・コールを呼ばないからである (表 3)。その結果、並列実行すると高い速度向上が望める。

4.3.2 make の並列実行

GNU make 3.77 を用い、Linux 2.0.36 のカーネル・コンパイルを同時に 2 つ実行させた。表 4 はその結果である。

make は複数のプログラムでジョブを構成しており、その中のプログラムの性質によって、ロック率も違ってくる。make の処理の大部分を占める cc1 は、字句解析、構文解析、中間言語の生成、上位レベルでの最適化という処理を行う。これは、ユーザ・レベルの処理が大部分である。ロック率は 1.41%である。これに対し、cpp はソース・ファイルを読み込んでマクロを展開することを行うため、ロック率は 21.4%になる。make の処理全体に対する平均のロック率は 5.11%である。

4.3.3 WWW サーバ

最近、WWW サーバとして、高いシェアを占めている Apache 1.3.6 について計測した。この WWW サーバは複数のプロセスを生成して、それらが、クライアントからの要求を処理するような並列プログラムである。SMP 型計算機で、クライアントからの要求を処理するプロセスを並列に動作させることができるのであれば、要求処理のスループットは 1 CPU の計算機より高いことが期待される。

実験にはサーバ側に複数個のファイルを用意しておき、サイズは 15KB、150KB の 2 種類にした。クライアントは、ひとつのプロセスで、はじめに指定回数だけ connect をして、サーバに接続し、それぞれ別のファイルを要求する。そして、すべてのファイルを獲得できたところで終了するようにした。クライアントには Celeron 366MHz の PC を用いた。また、サーバとクライアントの接続は 100BASE-TX のイーサネットである。

$$\text{スループット} = \frac{\text{クライアント数}}{\text{レスポンス+クライアント処理時間}} \quad \text{(要求数/秒)}$$

によりスループットを計算した。ネットワークの処理時間と比較してバッファにデータをコピーするなどのクライアントでの処理時間は 0 と仮定した。結果は、サーバが 2 CPU の場合は、1 CPU の場合よりむしろ性能が低下した (図 2)。

そこで、ロック率を調べた (表 5)。その結果、ロック率が 51.7%(150KB のとき) と非常に高い

ことが分かった。WWW サーバは要求ファイルを読む、それをネットワーク・デバイスを使って返す、というように共有資源を利用することが大部分の処理であり、`read` や `write` システム・コールが頻繁に呼ばれる。特に、Linux 2.0 では、システム・コールを呼ぶごとにロックがかかることになり、ロック率は非常に高くなる。

ロック率が高い並列プログラムはロック待ちの割合も高くなる。WWW サーバのロック待ちの割合は、30.4%(150KB のとき) であった。

実験結果から、WWW サーバのようにロック率が非常に高い並列プログラムを SMP 型計算機で動作させても、ほとんど性能向上しないことが明らかになった。つまり、複数のプロセスを独立に実行させても、実際には、並列度がないことになる。

5 おわりに

本論文では、ロック率に着目して並列プログラムを分類することを提案した。2 台の CPU を持つ SMP 型計算機上でロック率の異なる並列プログラムを測定した結果、ロック率が低い行列乗算プログラムでは高い性能向上が得られ、ロック率が高い WWW サーバではむしろ性能が低下することが確認された。ロック率に着目した並列プログラムの分類が有効であることが示された。

並列プログラムの設計者には、ロック率に着目してどれくらいの性能向上が得られるかを見積もることは大切である。また、SMP 型計算機の OS の設計者には、ロック率が小さくなるようなシステム機能の設計することが求められる。

本研究の今後の課題として以下のものが挙げられる。

1. 待ち行列網モデルによる性能予測手法との関係を考察する。
2. CPU を 3 台以上持つ SMP 型計算機で実験をする。
3. Linux 2.0だけではなく、Linux 2.2などロックのかけかたが違う OS で実験をする。
4. RAID など複雑な I/O システムを持つ計算機で実験をする。

参考文献

[1] G.Amdahl: Validity of the Single-processor Approach to Achieving Large Scale Computing Capabilities, *Proc. of AFIPS Conference*, pp.483-485 (1967).

[2] J.B.Chen, Y.Endo, K.Chen, D.Mazières, A.Dias, M.Seltzer and M.D.Smith: The Measured Performance of Personal Computer Operating System, *ACM Trans. on Computer Systems*, Vol.14, No.1, pp.3-40 (1996).

[3] 古市, 永松, 出口: 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).

[4] J.Gustafson: Reevaluating Amdahl's Law, *Comm. of the ACM*, Vol.31, No.5, pp.532-533 (1988).

[5] 堀川: ハイブリッド・モニタ手法を用いたシステム動作の測定・解析, 情報処理学会オペレーティング・システム研究会報告, 91-OS-50-10 (1991).

[6] 蔵杉, 海江田, 田中, 紀, 堀川, 中山: 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, 情報処理学会計算機アーキテクチャ研究会報告, 99-ARC-134-19 (1999).

[7] K.Lai and M.Baker: A Performance Comparison of UNIX Operating Systems on the Pentium, *Proc. of the USENIX 1996 Technical Conference*, pp.265-278 (1996).

[8] 小熊, 海江田, 森本, 田村, 鈴木, 中山: SMP 型計算機を活用する軽量プロセス・ライブラリ, 情報処理学会論文誌, Vol.39, No.9, pp.2718-2726 (1998).

[9] X.-H.Sun and D.T.Rover: Scalability of Parallel Algorithm-Machine Combinations, *IEEE Trans. on Parallel and Distributed Systems*, Vol.5, No.6, pp.599-613 (1994).