社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－ARC－171 (12)
2007－EMB－ 3 (12)
2007／1／23

デュアルコアプロセッサにおける最悪性能の確率的予測手法の提案

斉藤一樹 本田晋也 冨山宏之 高田広章
名古屋大学 大学院情報科学研究科
Email: { saitoh, honda, tomiyama, hiro } @ertl.jp

概要 近年，性能や低消費電力化の要求から，カーエレクトロニクスのようなハードリアルタイムシステムへのマルチコアプロセッサの利用が検討されている．ハードリアルタイムシステムの設計では，最悪実行時間を見積もることが重要であり，その方法として，静的解析に基づく手法が幾つか提案されている．しかし，その結果は過剰に悲観的であり，実際の設計に用いることは困難である．そのため，現実に適用可能な最悪実行時間を求めるために，確率的ハードリアルタイムという概念が提案されている．本論文では，この確率的ハードリアルタイムの概念をもちいて，マルチコアプロセッサ特有の実行時間の変動要因であるスピンロック待ち時間を予測する手法を提案し，実験によって提案手法の有用性の評価を行う．

Probabilistic prediction of WCET on Dual-Core Processors

Kazuki Saitoh Shinya Honda Hiroyuki Tomiyama Hiroaki Takada
Graduate School of Information Science, Nagoya Univ.
Email: { saitoh, honda, tomiyama, hiro } @ertl.jp

Abstract In recent year, the use of the Multi-Core processor from the demand of the performance and energy-saving to a Hard Real-Time system like the car electronics is examined. In the design of a Hard Real-Time system, it is important to estimate the Worst Case Execution Time(WCET), and is proposed some the techniques based on a static analysis as the method. However, it is difficult to use the result to design actually because it is excessively pessimistic. Therefore, to estimate the WCET that can be actually applied, the concept of Probabilistic Hard Real-Time is proposed. In this paper, it proposes the technique for predicting the spin-lock waiting time that is the change factor of a peculiar execution time to the Multi-Core processor by using this Probabilistic Hard Real-Time concept, and evaluated the utility of the proposal technique by the experiment.

1 はじめに

近年，家電製品・カーエレクトロニクスなどの組込み機器に要求される性能，機能は肥大化しており，要求されるプロセッサ性能も向上している．また，現在のシングルプロセッサ構成では動作周波数の増加による消費電力の向上と処理性能が限界に達することが予測される．そこで有力な解決法として，一つのチップに複数の CPU コアを内蔵したマルチコアプロセッサ化による処理性能の向上，低消費電力化が注目されている．

一方，カーエレクトロニクスのようなハードリアルタイムな時間制約を要求されるシステムにおいては，実行されるタスクは定められた時間内に実行を完了することが求められる．そのような時間制約を満たすシステムを設計するためには各タスクの最悪実行時間を知る必要がある．しかし，近年のプロセッサはパイプライン，キャッシュ，投機実行などの高速化手法の影響によってタスクの実行時間が変動するためシステムの設計が困難になっている．マルチコアプロセッサを用いた場合，それらに加えて排他処理の実現のためのスピンロック待ちと共有メモリのアクセス競合による待ち時間の影響が考えられる．

このような，変動要因の多いシステムの最悪実行時間を求める方法としては，長時間システムを動作させ，その中で観測された最長の実行時間を仮のタスク実行時間と見なし，安全のためのマージンを加えるという方法が一般的である．しかし，この手法ではプログラムを変更する度に毎回長時間の実行が必要となるため，膨大な時間がかかってしまう．

一方，最悪実行時間を理論的に求める手法として，シングルコアプロセッサ環境下では，プログラムソースの静的解析に基づいてタイミングスキーマを計算する手法 [1] や，静的解析と実行トレースによる動的な解析

を組み合わせた手法などが提案されている．しかしながら，これらの手法の結果は過剰に悲観的な値になることが多く，現実的なシステムの設計に用いることは困難である．その原因としては，個々の実行時間の変動要因すべてが最悪ケースである場合を想定していることが考えられる．しかし，このような状況が発生することはまれであり，例えば全てのメモリアクセスがキャッシュミスを引き起こすという状況は考えづらい．マルチコアプロセッサ特有の要因であるスピンロックやメモリアクセス競合の場合も同様に，最悪のケースが発生する確率は非常に低く，現実に起こることは実質的には無いと考えられる．

そこで，現実的なシステムの設計に用いることが可能な最悪実行時間を求めるための指標として，確率的なハードリアルタイムシステムという概念が提案されている [2]．これは，あるタスクに対する最悪実行時間をひとつの値として求めるのではなく実行時間の確率分布として求めるものである．確率的ハードリアルタイムに関する研究として，タイミングスキーマによる手法を拡張し，シングルコアプロセッサ上のタスクの実行時間の確率分布を求める手法 [2]，周期的リアルタイムシステム上のタスクプリエンプションをマルコフ過程と見做しデッドラインミス確率を求める手法 [3] などが提案されているが，マルチコアプロセッサに関する研究は行われていない．

本研究では，マルチコアプロセッサ上で動作するプログラムに対して，確率的ハードリアルタイムシステムの概念を適応した．具体的には，マルチコアプロセッサを用いたハードリアルタイムシステムにおいて，スピンロックによるロック取得待ち時間を，実行時間に対して最も大きな影響を及ぼす要因として着目し，短時間の実行から得た情報を用い，デュアルコアプロセッサで動作するタスクのスピンロック待ち時間の分布全体を推定する．

本論文の構成を以下に示す．まず，2 章ではマルチプロセッサ特有のタスク実行時間変動の要因について解説する．3 章では本手法の概要について説明し，4 章では提案手法で用いる統計学に関する計算方法，本手法の詳細な説明をする．5 章では提案手法を用いたロック待ち時間予測の実験を行い提案手法の有用性の評価を行う．

2　実行時間の変動要因

シングルコアプロセッサにおける実行時間の変動要因としてパイプライン，キャッシュ，投機実行などが挙げられる．一方，マルチコアプロセッサ特有の要因としては，メモリアクセス競合とスピンロック待ちがある．ここで，メモリアクセス競合による実行時間の遅延は，スピンロックによる遅延ほど大きな影響は無いと考えられる．そこで，本論文ではスピンロックによるロック待ち時間のみに着目する．

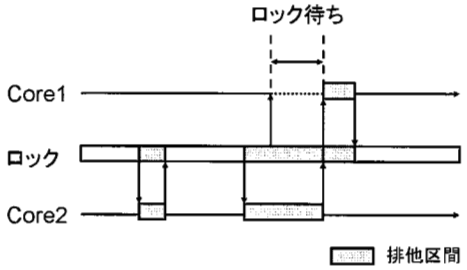

図 1　スピンロックによるロック待ちの発生例

一般的なマルチコアプロセッサ対応のソフトウェアにおいて，コア間の排他制御は，ロック変数を用いたスピンロックによって実現できる．あるプロセッサコアで実行中のタスクが排他処理を行う場合，ロックを取得した後行いたい処理を実行し最後にロックを解放する．ロック取得の際，他のコアが既に排他区間に入っていた場合ロック取得の競合が発生しロックの取得は失敗する．この場合，タスクは他のコアがロックを解放するまでの間，ビジーループによって待たされることになる (図 1)．ロック待ち時間はロック取得を試みるタイミングによって様々に変わるため，システムを設計する際，ロック待ち時間を何らかの方法で決定する必要がある．唯一つのロックによって全ての共有資源の排他制御を実現しているとき，そのロック変数をジャイアントロックと呼ぶ．

デュアルコアプロセッサの場合，あるタスクのロック待ち時間の考え得る最悪の値は，タスクの排他区間数と他コアで実行し得る最長の排他時間の積によって求めることが出来る．しかし，ロック取得を試みる時刻と相手コアが実行し得る最長の排他区間の開始時刻が毎回一致するという状況は，現実に発生するとは考えづらい．そこで，本手法ではこの問題に対し，確率的ハードリアルタイムシステムという概念を適用し，次章で説明するモデルにおけるロック待ち時間の確率分布の予測を行う．

3　手法の概要

本手法で用いるシステムのモデルについて説明する．予測の対象とするシステムは，デュアルコアプロセッサ上で動作しており，全てのタスクは一定時間ごとの周期割り込みによって起動する周期タスクで構成され

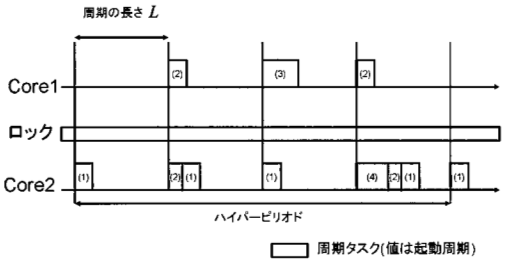

図 2　システムのモデル

ている．また，タスクのプリエンプション，割り込み，外部入出力は発生しない．ロックはジャイアントロックのみであり，複数のロックを取得する場合は考えない．各タスクは毎回同じ処理を行う．そのため，排他区間の数，長さは常に同じである．以下，説明のために予測対象のタスクが動いているコアをコア 1，もう一方のコアをコア 2 と呼ぶ．図 2 にシステムのモデルを示す．

次に，本手法の概要を説明する．提案する手法は，予測対象のシステムを動作させた結果を用いるというアプローチである．システムを動作させる際は，各コアごとに一方のみを動作させた実行ログを取る．実行ログを取る動作時間は，コア 1 に対しては予測対象タスクが起動する 1 周期分，コア 2 に対しては時刻 0 からハイパーピリオドまで実行させる必要がある．ハイパーピリオドとは，全ての周期タスクの周期の最小公倍数であり，ハイパーピリオド毎に同じスケジュールが繰り返される．ロック待ち時間の確率分布の予測にはまず，ロック取得の競合回数の確率分布を予測する．次に，求めた競合回数の確率分布を用いて対象タスクの 1 回の実行におけるロック取得待ち時間の確率分布の予測を行う．

4　最悪性能の確率的予測

本章ではまず，4.1 節で提案手法で用いる確率分布関数に対する演算である畳み込み演算について説明し，4.2 節でロック取得の競合回数の確率分布の予測手法について，4.3 節ではロック取得待ち時間の確率分布の予測手法について説明する．

4.1　畳み込み

2 つの互いに独立な確率変数 X, Y があり，それらの確率分布関数を $g(x), h(y)$ とする．この 2 つの確率分布の和 $Z = X + Y$ の分布 $k(z)$ を考える．$X + Y = z$ となるのは $X = x, Y = z - x$ を満たす全ての組み合わせであるからそれらを確率の積で以下の式で表すことが出来る．

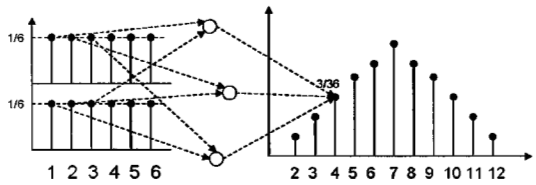

図 3　畳み込みの例

$$k(z) = \sum_x g(x)h(z-x) \tag{1}$$

この演算を畳み込み演算と言い，以後，$k = g \otimes h$ と表記する．

畳み込みは以下の性質を満たす演算である．

1. 交換律 $g \otimes h = h \otimes g$
2. 結合律 $(g \otimes h) \otimes k = g \otimes (h \otimes k)$
3. 分配律 $(g \otimes h) + z = (g \otimes z) + (h \otimes z)$
4. スカラー倍 $a(g \otimes h) = (ag) \otimes h = g \otimes (ah)$

また，ある確率分布関数に対して式 2 で与えられる確率分布関数 δ を畳み込み演算に作用させた結果は元の確率分布関数と同じものになる．

$$\delta(t) = \begin{cases} 1 & (t = 0) \\ 0 & (else) \end{cases} \tag{2}$$

図 3 に 2 つのサイコロを投げて出た目の和の確率分布 $k(z)$ を求める例を示す．サイコロ 1 つを投げたときの目の確率分布 $g(x)$ とすると求める確率分布は $k(z) = g(x) \otimes g(y)$ によって計算できる．

目の和が 4 となる確率 $k(4)$ を求める場合を考える．目の和が 4 になる組み合わせは $\{1,3\}\{2,2\}\{3,1\}$ の 3 つなので $k(4) = g(1) \times g(3) + g(2) \times g(2) + g(3) \times g(1)$ となる．

4.2　ロック取得の競合回数の確率

ロック待ち時間の予測を行う前段階として，まずロック取得の競合回数の確率分布を求める．

もし，コア 2 の実行の間，排他処理をしている割合が実行中常に一様であれば，一定の確率でランダムにロック取得の競合が発生すると考えられる．競合発生確率 p は排他処理を行っている割合と同じなので，実行ログから排他時間の合計を計算し，その値を実行時間で割ることで求めることができる．競合回数の確率分布は予測対象タスクの排他区間数 n，競合回数 x として以下の二項分布 $Bi_{n,p}(x)$ で表すことができる．

$$Bi_{n,p}(x) = {}_nC_x p^x (1-p)^{n-x} \tag{3}$$

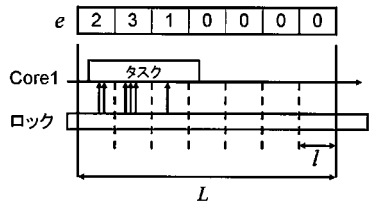

図 4　排他処理の開始タイミング e

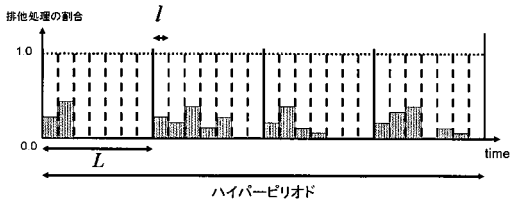

図 5　コア 2 の排他処理の割合 f

しかし，実際のソフトウェアでは排他処理を行う割合はタスクによって異なるため，排他の割合が常に一定とは言えない．そのため，実行中コア 2 の排他の割合が変化する場合を考慮する必要がある．4.2.1 節，4.2.2 節では排他の割合の変化を考慮した手法について説明する．

4.2.1　予測に用いる情報

排他の割合が変化する場合を考慮した手法を用いるため，2 つの配列 e, f を求める．配列 e は予測対象タスクが排他処理に入るタイミングを表す．このデータを得るためには，まずコア 1 の実行ログのある 1 周期 L を一定の幅 l ごとの区間に区切る．ここで，区間の分割幅 l は要求される予測の精度によって適切に決める必要がある．次に，要素数 L/l の配列を用意し，値を全て 0 に初期化する．この配列の各要素は分割した周期 L の各区間に対応する．予測対象タスクが排他処理に入った時刻が分割後のどの区間に該当するかを求め，その区間に対応する配列要素を 1 加算する．配列 e を求めた例を図 4 に示す．

配列 f は，コア 2 の実行中の排他処理の割合を表す配列である．これを得るためには，コア 2 の実行ログ中のあるハイパーピリオドに対して，e を求めたときと同様に，幅 l を用いて N 個の区間に区切り，配列 f を用意する．この配列の要素はそれぞれ各区間に対応し，区間ごとの排他処理を行っている割合を表す．配列 f の例を図 5 に示す．

4.2.2　ロック競合回数の予測

幅 l で分割したある区間 i における競合回数の確率分布は，その区間内でコア 2 が排他処理をしている割合を一定であると見なすことで，単純な二項分布によって求めることが出来る．全ての区間に対する競合回数の確率分布を求めた後，それら全ての分布の和の分布を求めることで，タスクの起動から終了までに発生する競合回数の確率分布を求めることが出来る．

ここで，二項分布の形状を決定するパラメータとして，区間 i における競合発生確率 f_i と区間 i 中の予測対象タスクの排他区間数 n_i が必要となる．f_i はコア 2 の排他処理の割合を示す配列 f の区間 i に対応する要素である．しかし，n_i は予測対象タスクの起動タイミングによって様々に変化する．そこで，本手法では全てのタスク起動タイミングに対してタスクの競合回数の確率分布を求め，それらの平均を求めたものを競合回数の確率分布とする．以上の手順を以下にまとめる．

1. $j \leftarrow 0$
2. $i \leftarrow j \sim j + L/l$ に対し，二項分布 $Bi_{e_{i-j},f_i}(x)$ を求める
3. $P_j(x) \leftarrow Bi_{e_{0-j},f_0}(x) \otimes Bi_{e_{1-j},f_1}(x) \otimes ... \otimes Bi_{e_{L/l-j},f_{L/l}}(x)$
4. $j = N$ ならば 5 へ，そうでなければ j に 1 加算して 2 へ
5. $P_{contension}(x) \leftarrow \frac{1}{N-1} \sum_{j=0}^{N-1} P_j(x)$

ここで，$P_{contension}(x)$ が求めるべきロック競合回数の確率分布である．

4.3　スピンロック待ち時間の分布

スピンロック待ち時間の分布を求めるために，まず競合 1 回あたりの待ち時間の確率分布を考える．コア 2 の排他処理にかかる時間が全て同じ長さ t である場合，予測対象タスクは 0 ～ t のいずれかの時刻に一様の確率でロック取得を試みると考えられる．そのため，ロック待ち時間の分布は $[0, t]$ 上の一様分布となる．

排他時間が様々である場合，コア 2 の実行ログのあるハイパーピリオドから，排他区間の長さの分布を求める必要がある．排他区間長の分布は，コア 2 の排他処理を処理時間ごとに分類し，集計したもので，排他区間長の種類が M 個あるとき 2 つの値の対 $\{m_i, p_i\}$ の M 個の配列で表される．ここで m_i は排他処理にかかった時間，p_i は全排他区間数に対する排他時間が m_i である排他区間数の割合である．$\{m_i, p_i\}$ の対を，常に $m_i \leq m_{i+1} (1 \leq i \leq M-1)$ となるように整列したものを排他区間長の分布として用いる．

排他区間長の分布を用いて，1 回競合した場合のロック待ち時間の分布 $P_1(t)$ は以下の式によって求められる．

$$P_{delay,1}(t) = \begin{cases} \frac{p_1}{m_1} + \frac{p_2}{m_2} + ... + \frac{p_n}{m_n} & (0 \leq t < m_1) \\ \frac{p_2}{m_2} + ... + \frac{p_n}{m_n} & (m_1 \leq t < m_2) \\ ... & \\ \frac{p}{m_M} & (m_{M-1} \leq t \leq m_M) \end{cases} \quad (4)$$

予測対象タスクにロック取得の競合が x 回発生したときのロック待ち時間の確立分布 $P_{delay,x}(t)$ は，$P_{delay,1}(t)$ に従う独立な x 個の確率変数の和の分布であるため，畳み込み演算を用いて以下の式で表すことができる．

$$P_{delay,x} = P_{delay,x-1} \otimes P_{delay,1} \quad (5)$$

求めるべきロック待ち時間の分布は 4.2 節にて求めた競合回数の確率分布 $P_{contension}$ を用いて以下の式で表すことが出来る．

$$P_{delay}(t) = \sum_{i=0}^{n} (P_{contension}(i) \times P_{delay,i}(t)) \quad (6)$$

ここで n は予測対象タスクが実行する排他区間の数を表す．

5 提案手法の実験と評価

本手法を用いてロック待ち時間の確率分布を予測し，実際の測定値から得られた分布と比較することで，提案手法の有用性の評価を行う．実験に用いるシステムは，エンジン制御用ハードウェアのベンチマークに用いられているソフトウェアをデュアルコア上で動作するよう 2 つに分割したものである．両コアとも周期割り込みによっていくつかのタスクが起動し，全てのタスクが必要な処理を終えると次の周期割り込みまでアイドルタスクが実行される．アイドルタスクはハードウェアのチェックを行うだけの最も優先度の低いタスクであり，このタスク内で排他処理が実行されることは無い．また，システムに対する外部入出力は省かれており，周期割り込み以外の割り込みは発生しない．予測対象のタスクはコア 1 で動作しており，$1ms$ ごとの周期割り込みによって起動し毎回同じ処理をして終了するものを選んだ．このタスクは起動から終了の間に 6 回排他処理に入る．

実験はコア 2 のロック取得頻度の性質が異なる 2 種類の設定に対して行った．5.2 節ではコア 2 の排他処理を実行する頻度を実行中常にほぼ一定にするために，コア 2 の割り込み周期を短く設定し，アイドルタスクが起動しないようにした．5.3 節では排他の頻度に偏りが出るようにコア 2 の割り込み周期を長く設定し，アイドルタスクが起動するようにした．図 6，図 8 はそれぞれの実験のコア 2 のある 1 周期におけるロック取得割合の分布を示したものである．

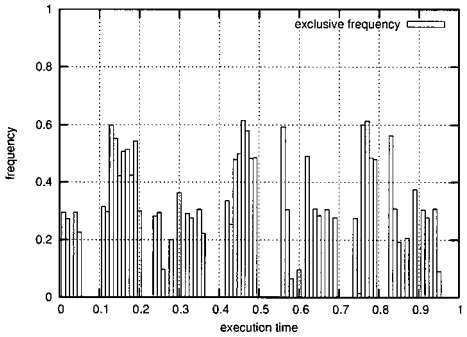

図 6　コア 2 の排他の頻度

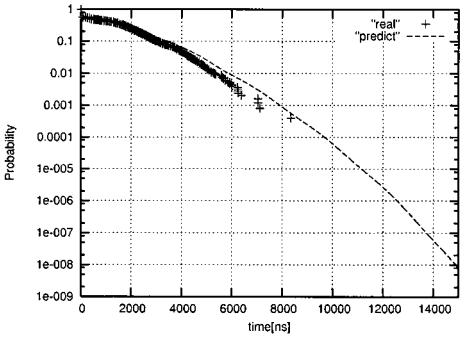

図 7　ロック待ち時間の確率分布の予測結果

5.1 ロック待ち時間の測定方法

実験はどちらも SystemC によって構成されたシミュレータ上で上記ソフトウェアを動作させ，5000 回ロック待ち時間の測定を行った．測定の際は毎回，予測対象タスクの起動時刻を一定間隔ずつずらすことで，無作為に様々なロック取得の競合のパターンを発生させた．ロック待ち時間はロック取得を試みるタイミングによって無数のパターンが考えられるが，無作為にタイミングを決めることで，全ての競合のパターンの傾向を良く反映したロック待ち時間の分布が得られると考えられる．

5.2 実験 1

コア 2 に実行中常にほぼ同じ頻度で排他処理を実行させた場合の実験結果を図 7 に示す．グラフは横軸にロック待ち時間をとり，縦軸に確率を対数で表している．確率は 1 から累積確率を減算した値であるため，ある待ち時間 t に対する確率は，待ち時間が t 以上である確率を表している．" real" は実際に観測されたロック取得待ち時間の分布である．"predict" はロック取得の競合回数の確率を単純な二項分布として予測した結果である．

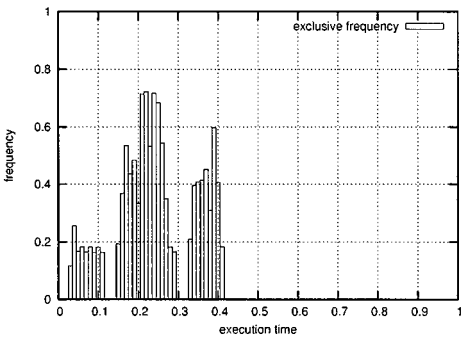

図 8　コア 2 の排他の頻度

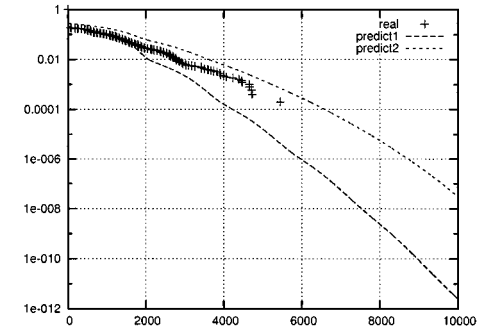

図 9　ロック待ち時間の確率分布の予測結果

コア 2 の排他区間のプロファイルから得られる最長の排他時間は $4500ns$ であるため，考え得る最悪のロック待ち時間は $27000ns$ なのに対して，より現実的な結果が得られていると考えられる．

5.3　実験 2

コア 2 の実行中のロック取得頻度に偏りがある場合の結果を図 9 に示す．ここで，"predict1" は 5.2 節同様，ロック取得の競合回数の確率を単純な二項分布として予測した結果であるが，ロック待ち時間 $4000ns$ の時点で既に実際に観測された確率を大きく下回っており，その後も誤差は広がり続けている．"predict2" は排他区間の分布の偏りを考慮した手法を用いた予測結果である．こちらは"predict1" に比べて誤差は少なく，グラフの中央付近に来ても誤差はそれほど拡大しておらず，よく予測できていると言える．予測のためのパラメータとして $l = 100000[ns]$ を用いた．

6　まとめと今後の課題

本論文では確率的ハードリアルタイムシステムの概念について紹介し，統計学的な前提の下にデュアルコアプロセッサにおけるロック待ち時間の確率分布を予測する手法を提案した．さらに，実際に使われている大規模なソフトウェアを用いて提案手法の有用性を確認する実験を行った．

今回提案した手法では，4.2.1 節で配列 e を求める際，ある 1 周期における実行ログのみを用いた．しかし，排他処理開始のタイミングはスピンロック，キャッシュ，パイプラインなどの影響によって様々に変化するため，e の各要素の値も実行ごとに変化する可能性がある．予測と実測との誤差はそのためであると考えられる．この点を改善するためには，それぞれの要因に対する予測手法を組み合わせることで配列 e を求める必要がある．

今後の課題として，本手法のロックが複数ある場合，3 コア以上のマルチコアプロセッサである場合への適用，メモリアクセス競合による遅延時間の予測などが挙げられる．

謝辞

本研究は共同研究の一環として行われた．トヨタ自動車，NEC エレクトロニクスをはじめ，研究にご協力くださった関係者各位に厚く御礼申し上げます．

参考文献

[1] C.Park, A.Shaw, "Experiments with a program timing tool based on source-leel timing schema", IEEE Transactions on Computers, 1991

[2] Guilliem Barnat,Antoine Colin,Stefan M.Petters, "WCET Analysis of Probabilistic hard Real-Time System", IEEE RTSS'02,2002.

[3] Jose Luis Diaz,Daniel F.Garcia,Kanghee Kim,Chang-Gun Lee,Lucia Lo Bello Jose Maria Lopez,Sang Lyul Min,Orazio Mirabella, "Stochastic Analysis of Periodic Real-Time Systems",IEEE RTSS'02,2002.