3N-04　スレッドの稼働履歴に基づくリアルタイムスケジューリング技法

勝部 弘嗣 †　谷出 新 ††　瀧本 栄二 ††　芝 公仁 ††　大久保 英嗣 †
† 立命館大学理工学部情報学科　†† 立命館大学大学院理工学研究科

1　はじめに

近年, マルチメディアの普及により, 連続メディアデータの再生・転送処理などの実時間処理を行うシステムの重要性が高まっている. このようなシステムでは, オペレーティングシステム（以下, OS）が実時間処理を行うための機構を提供することが望まれる.

リアルタイム OS をマルチメディアに適用する場合, 実行するスレッドは静的に決定できないため, 動的なスレッド生成に対応する必要がある. リアルタイムシステムでは, スレッドを生成しても, 他のスレッドが時間的制約を満たすようにしなければならない. そのため, スレッド生成の要求を受けたカーネルは, スケジューリング可能性判定を行い, スケジューリング可能であると判定した場合のみ, スレッドを生成するという手法をとる.

本稿では, スレッドの稼働履歴に基づいたスケジューリング可能性判定を行う手法について述べる. 本手法により, CPU 資源を有効に利用したリアルタイムスケジューリングを行うことが可能となる.

以下, 2 章では CPU 利用率算出手法, 3 章では本機構の評価について述べる. 最後に, 4 章で本稿のまとめについて述べる.

2　CPU 利用率算出手法

スケジューリング可能性判定を行うためには, 現状でどのくらいの CPU 資源を消費しているかを把握する必要がある. 本章では, 従来の手法による算出方法の欠点と提案手法における算出方法について述べる.

2.1　従来の CPU 利用率の算出手法

従来の手法では, 現時点でスケジューリングの対象となっているリアルタイムスレッドの CPU 利用率を合計することにより, 現時点での CPU 利用率を求める. すなわち,n を全周期スレッド数, i 番目のスレッドの実行時間と起動周期をそれぞれ C_i と T_i としたとき, システムの CPU 利用率は, 式 (1) となる [1].

$$CPU\text{ 利用率 }(\%) = \sum_{i=1}^{n} \frac{C_i}{T_i} \times 100 \qquad (1)$$

多くの場合, 実行時間には WCET(Worst-Case Execution Time: 最悪実行時間) が用いられるが, 実際の実行時間が WCET と等しくなることは少ない. また, スレッドの実行時間は, 実行環境に大きく依存するため, 単一の WCET を用いた場合, 想定している CPU 利用率と実際の CPU 利用率に差が生じ, 結果として CPU 資源を有効に利用できなくなることが考えられる.

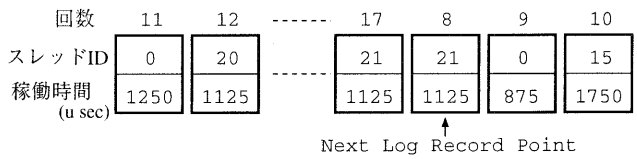

図 1 ログデータの概念図

2.2　履歴に基づいた CPU 利用率の算出手法

従来の手法に対し, 我々が提案する手法では, スレッドの稼働状況を基に CPU 利用率の算出を行い, 得られた CPU 利用率を用いてスケジューリング可能性判定を行う. これにより, 従来の手法よりも, 実状に合致したスケジューリングが可能となり, CPU 資源を有効に利用することができる.

本手法では, スレッドの切り替えが発生するたびに, それまで稼働していたスレッドのスレッド ID と稼働時間をログに記録する. 稼働時間は, スレッドが実行状態になった時刻と次のスレッドが実行状態になった時刻との差である. CPU 利用率は, ログデータの稼働時間の合計に対するアイドルスレッドの稼働時間の割合から算出される.

図 1 に, 過去 10 回分のデータを保持し, 17 回目のログを保存した後のログデータの概念図を示す. このようにログデータ全体のサイズは固定であり, 古いものから順に上書きされる. 図中の Next Log Record Point は, 次にログを保存する場所を示す. このときの CPU 利用率は, ログデータの合計時間を T, アイドルスレッドの稼働時間の合計を I とすると, 式 (2) で求めることができる.

$$CPU\text{ 利用率 }(\%) \quad = \quad (1 - \frac{I}{T}) \times 100 \qquad (2)$$

A Real-time Scheduling Method based on the Execution Log of Threads

Hirotsugu Katsube†, Hajime Tanide††, Eiji Takimoto††, Masahito Shiba††, and Eiji Okubo†
†Department of Computer Science, Faculty of Science and Engineering, Ritsumeikan University
††Graduate School of Science and Engineering, Ritsumeikan University

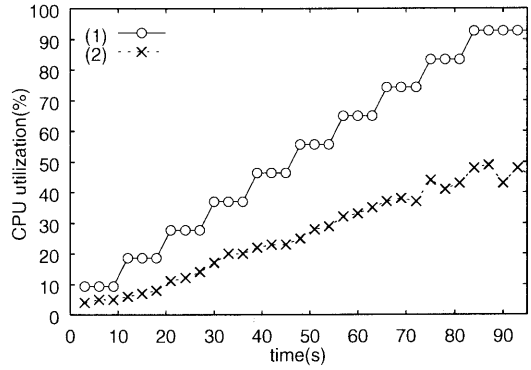

図 2 CPU 利用率算出結果（ランダム）

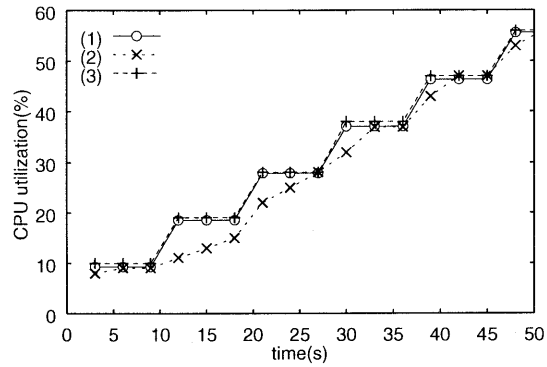

図 3 CPU 利用率算出結果（WCET）

3 評価

本機構を, 我々が開発しているリアルタイム OS Easel に実装し, 評価を行った. 実験は, MMX Pentium 200MHz の PC/AT 互換機を用いて行った.

本機構の特性を表すグラフを図 2, 3 に示す. 図 2 は, 周期 100ms, デッドライン 100ms, WCET9.25ms で実行時間が一様分布になるようなスレッドを 10 秒ごとに 1 つずつシステムに追加したとき, それぞれの機構が算出した CPU 利用率をグラフに表したものである. 従来の手法で算出を行った場合, WCET をもとに固定的に算出を行うため, スレッドを投入した直後に値が反映され (1) のようなグラフとなる. 一方, 本機構を用いた場合, 実際のスレッドの稼働時間から算出を行うため, (2) のようなグラフとなる. 時刻 90 の時点で, 従来の算出方法では 92.5%となっているが, 実際の CPU 利用率は 50%程度である.

図 3 は, 図 2 と同じ属性のスレッドで, 実行時間が常に WCET となるようにし, 図 2 と同じように実験を行った結果である. (1) は従来の手法による算出の結果で, スレッドの実行時間とは関係なく算出を行うため, 図 2 の (1) と同じグラフとなる. また, 実行時間は常に WCET であるため, (1) のグラフが理想のグラフとなる. 一方, 本機構により算出を行った結果は, (2) のように滑らかなグラフとなり, システムの変化に対して反応が遅くなっている. 本機構のログデータは, スレッドの切り替え時に記録されるため, スレッドの切り替わりが少ない場合, ログデータの稼働時間の合計は長くなる. そのため, ログデータの合計時間に対して, 新たに生成されたスレッドの稼働時間が小さいため, あまり CPU 利用率に反映されなくなってしまう.

表 1 追加可能なスレッド数

	従来手法	提案手法
Random	10	20
WCET	10	10

(3) のグラフは, 過去 1 秒間のログデータを用いて算出を行ったものである. この手法では, 理想のグラフとほぼ同じ軌跡を描いていることから, 十分に実用可能であることがわかる. (3) のグラフが, (1) よりも高い値を示しているのは, ユーザが指定する WCET は, 純粋なスレッドの実行時間であり, スケジューリングオーバヘッドの時間を含まないことによる. 本機構では, これらのオーバヘッドもスレッドの実行時間として計測されるため, 理想のグラフよりも高い値を示す.

次に, 従来の CPU 利用率の算出方法と本機構の CPU 利用率の算出方法で, それぞれ CPU 利用率の上限を 100%とし, 同じ特性を持つスレッドをいくつ追加できるかという実験を行った. スレッドは, 先程の 2 種類のスレッドを用いた. その結果を表 1 に示す.

スレッドの実行時間が一様分布になるとき, スレッドの実行時間の平均は約 4.6ms となる. すなわち, スレッド 1 つの平均 CPU 利用率は 4.6%といえる. 従来の算出方法では, 固定的に CPU 利用率を 9.25%として判断してしまうため, スレッドを 10 個実行した時点で CPU 利用率は 92.5%となり, それ以上スレッドを追加することができなくなる. しかし, 実際の CPU 利用率は 46%程度であり, CPU 資源にはまだ余裕があることになる. 本機構では, 実測に基づき算出を行うため,10 個以上の追加が可能となる. 今回の実験では, 20 個まで追加することができた.

4 おわりに

本稿では, スレッドの稼働履歴に基づいた CPU 利用率算出機構について述べた. 本機構では, CPU 利用率を実測により算出するため, 現状でのシステム負荷を考慮したサービスが可能となり, 従来の手法に比べ, CPU 資源を有効に活用したリアルタイムスケジューリングを行うことが可能となる.

参考文献

[1] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real time environment," J. ACM, Vol. 20, No. 1, pp. 46–61 (1973).