B-013

余裕時間によって動作を切り替える EDZL に基づいたアルゴリズムの提案と可能性解析

A scheduling algorithm based on EDZL that switches operation depending on laxity time and analysis of schedulibility

上向 翔太† 俞 明連† 中村 徹†
Shota Uemukai Myungryun Yoo Toru Nakamura

1. はじめに

プリケーションやロボットの出現により，高性能なプラッ トフォームを必要としている．組込システムには消費電力の制約があり，プロセッサ自体の性能を上げることが難しくなってきているためマルチプロセッサ技術による性能の向上，リアルタイム性の保証が重要である.[1] また，その中でもマルチプロセッサ環境下において効率の良いタスク処理のスケジューリングが求められている．しかしシングルプロセッサ環境下におけるタスク処理のスケジューリング方法である RM(Rate Monotonic)[2] や EDF(Earliest Deadline First)[2] はマルチプロセッサ環境では最適ではないと知られている．マルチプロセッサ 環境下で CPU を 100%利用できるアルゴリズムも提唱されている[3][4]が，複雑でかつコンテキストス イッチも多いため実用には至っていない．そこで EDF を基にしたマルチプロセッサ環境用のスケジューリングアルゴリズムに EDZL(Earliest Deadline Zero Laxity) [5]が提案されたがまだ改善点を抱えている．

1.1 システムモデル

本研究では,周期的タスクからなるタスクセットのスケジューリングを対象とし,以下のシステムとして定義する．また本論文で用いる記号を表 1 にまとめる．

表 1.各記号とその意味

記号	意味
M	プロセッサ数
N	タスク数
τ	タスクセット
$\tau_i(\tau_i \in \tau)$	タスクセット内の i 番目のタスク
C_i	τ_iの最悪実行時間
T_i	τ_iの周期
U_i=C_i/T_i	τ_iのタスク利用率
$U = \Sigma_{i=1}^{N} U_i$	システム全体の負荷
U/M	システム利用率

システムはM個のプロセッサを持ち，タスクセットτは各タスク$\tau_i(i = 1,2,\dots,N)$によって構成されている，すなわち$\tau = \{\tau_1, \tau_2, \dots, \tau_N\}$である.また各タスクは独立しているとする.すなわち各タスクに実行順序などの依存関係はないものとする．各タスクはプリエンプト可能であるがどのようなタスクも複数のプロセッサで同時に実行することはできないものとする．

各タスクτ_iは$\tau_i = (C_i, T_i)$と定義され,その利用率は$U_i = C_i/T_i$とする．あるジョブのデッドラインは次のジョブのリリース時刻となる．

† 東京都市大学 Tokyo City University

1.2 関連研究

1.2.1 EDF(Earliest Deadline First)

EDF[2]はスケジューラがタスクの起動時と完了時に起動し、デッドラインの早いジョブほど高優先度が与えられるアルゴリズムである．図 1 はあるタスクセットをプロセッサ数 2 の EDF で処理した場合の実行例である．この場合のデッドラインオーバー量は 4, スケジューラ起動回数は 4 回となる．

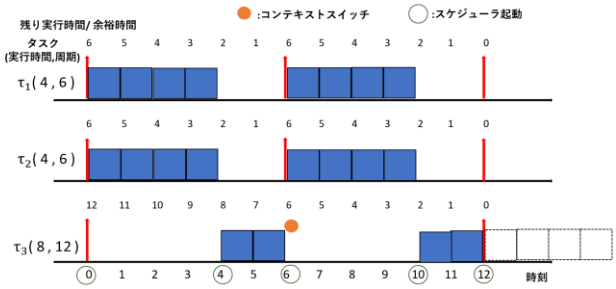

図 1 EDF の実行例

1.2.2 EDZL(Earliest Deadline Zero Laxity)

EDZL[5] はスケジューラが毎単位時間起動し，デッド ラインの早いタスクほど高優先度が与えられるが，余裕時間が 0 になったタスクに最高優先度が与えられるスケジューリングアルゴリズムである．余裕時間とはあるタスクの現在時刻からデッドラインまでの時間から残り実行時間を引いたものであり，これがゼロになったときにそのタスクが速やかに実行されなければデッドラインミスを起こすということを意味している．

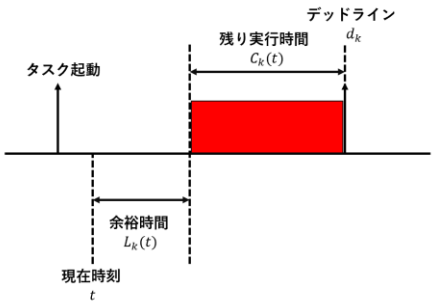

図 2 余裕時間の概念図

プロセッサ数を 2 とし，図 1 と同じタスクセットを対象に EDZL で実行した場合の実行例は図 3 のようになり，この場合のデッドラインオーバー量は 2,コンテキストス イッ

Copyright © 2024 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

チ回数は 2 回,スケジューラ起動回数は 12 回となり， EDF に比べデッドラインオーバー量が減ったがス ケジューラが毎単位時間起動するためオーバーヘッドが多いという問題点を持つ.

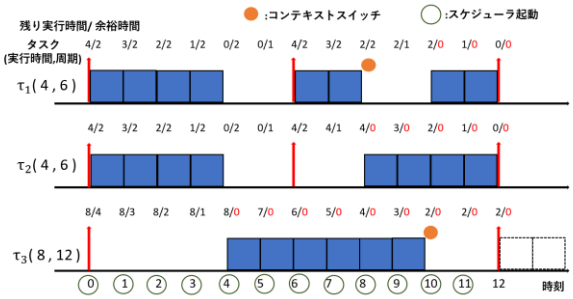

図 3 EDZL の実行例

2. 提案手法

1.2 で述べた関連研究から本研究では EDF と EDZL を余裕時間がクリティカルなタスクが存在するかで切り替えるアルゴリズムを提案する. 余裕時間がクリティカルとは,そのタスクの余裕時間が最もデッドラインの早いジョブの最小の残り実行時間よりも小さいことを指す.

余裕時間がクリティカルなタスクがない間はスケジューラはタスクの起動時と完了時とし， デッドラインが早いタスクを優先実行する. 余裕時間がクリティカルなタス クがある間は毎単位時間スケジューラが起動し, 余裕時間が 0 のタスクを最優先, 次にデッドラインが早いタスクを優先的に実行する. プロセッサ数を 2 とし, 図 1 と同じタスクセットを提案手法で実行した場合の実行例は図 4 のようになり, この場合のデッドラインオーバー量 2, コンテキストスイッチ回数は 2 回, スケジューラ起動回 数は 9 回となる.

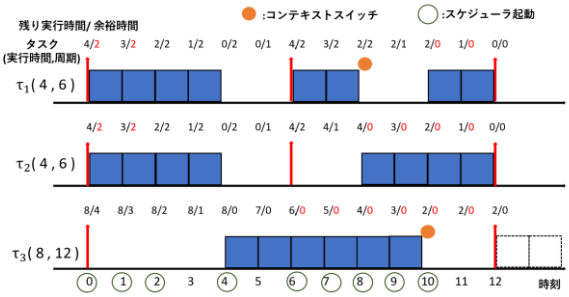

図 4 提案手法の実行例

3. 可能性解析と評価

3.1 可能性解析

スケジュール可能性解析とは与えられたタスクセットがスケジュール可能か否かを判定することのできる条件を求めることである. 本研究ではタスクの反応時間解析 (Response Time Analysis)[6]に基づいた方法で行う.

そのためにここで 2 つの用語を定義する.

定義 1(干渉長). I_k を区間 [a,b)においてτ$_k$がプロセッサ数以上の高優先度タスクによってブロックされている長さの合計を, I_k^iはτ$_i$がブロックされて実行できない区間の長さを表す.

定義 2(仕事量) W_iはタスクτ$_i$が区間 [a,b)において実行しなければならない実行量を指す.

また干渉長に関する補題を以下に示す.

補題 1 $I_k(a,b) = \Sigma_{i=1}^{N} I_k^i(a,b)/M$が成立する.

Proof. グローバル方式のスケジューリングアルゴリズムは仕事量を保存するためジョブが実行可能であるのに実行されないとき, 各プロセッサは必ずほかのタスクのジョブにより占有されているということは明らかである.

補題 2 すべてのグローバルスケジューリングアルゴリズムにおいて以下の式が成立する.

$$I_k(a,b) \geq x \Leftrightarrow \Sigma_{i\neq k} \min\left(I_k^i(a,b), x\right) \geq Mx \quad (1)$$

Proof. εを$I_k^i(a,b) \geq x$であるタスクの個数としたとき,

$$\begin{aligned}\Sigma_{i=1}^{N} \min\left(I_k^i(a,b), x\right) &= \varepsilon x + \Sigma_{i:I_k^i<x} I_k^i(a,b) \\ &= \varepsilon x + MI_k(a,b) - \Sigma_{i:\geq I_k^i x} I_k^i(a,b) \\ &\geq \varepsilon x + MI_k(a,b) - \varepsilon I_k(a,b) \\ &= \varepsilon x + (M-\varepsilon)x \\ &= Mx \end{aligned} \quad (2)$$

ここで$\Sigma_{i=1}^{N} \min\left(I_k^i(a,b), x\right) \geq Mx$とすると,

$$\begin{aligned} I_k(a,b) &= \Sigma_{i=1}^{N} I_k^i(a,b)/M \\ &\geq \Sigma_{i=1}^{N} \min\left(I_k^i(a,b), x\right)/M \\ &\geq Mx/M \\ &= x \end{aligned} \quad (3)$$

解析の手順を以下に示す.各タスクτ$_k$の反応時間が最大となる状況を考える. その時のジョブをJ_k^*とし,最大反応時間をR_k^{ub}とする. ジョブJ_k^*について区間$[r_k^*, R_k^{ub})$におけるτ_kへの干渉長I_kの上限値I_k^{ub}をもとめ, 干渉長の上限値I_k^{ub}とτ_kの実行時間からτ_kへの反応時間の上限値R_k^{ub}を求める. R_k^{ub}を求めることができれば各タスクのすべてのジョブについて他のタスクからの干渉が最も大きいジョブが実行を完了する時間(最大反応時間)を求めることで各タスクがデッドラインミスを起こすかどうかを判定できる. また解析のために 2 つの補題を示す.

補題 3 区間[a,b)においてτ$_i$によるτ$_k$への干渉長$I_k^i(a,b)$は仕事量W_i(a,b)を超えない.

Proof. タスクは実行しているときのみほかのタスクに鑑賞できるので干渉長と仕事量の定義により明らかである.

補題 4 タスクτ$_i$の反応時間は,下式を満たせばR_k^{ub}を超えない.

$$\Sigma_{i\neq k} min\left(I_k^i\left(R_k^{ub}\right), \left(R_k^{ub} - C_k + 1\right)\right) < \left(R_k^{ub} - C_k + 1\right) \quad (4)$$

Proof. 式が成り立つとすると補題 2 より以下の式が成り立つ.

$$I_k\left(R_k^{ub}\right) < \left(R_k^{ub} - C_k + 1\right) \quad (5)$$

Copyright © 2024 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

よって$R_k^{ub} - C_k$よりも長い干渉は受けない.したがって干渉長の定義によりR_k^{ub}までに実行を完了することができる.

以上により各タスクτ_kの反応時間の上限値R_k^{ub}を求めるため他のタスクτ_iによるτ_kへの干渉長を求める必要がある.マルチプロセッサシステムにおけるτ_iがデッドラインミスを起こさない場合の最悪の仕事量の上限値は以下の式で計算できる.

$$N_i(R_k^{ub}) = \left\lfloor \frac{R_k^{ub} - C_i + T_i}{T_i} \right\rfloor \quad (6)$$

$$\omega_i(R_k^{ub}) = N_i(R_k^{ub})C_i + \min(C_i, R_k^{ub} - N_i(R_k^{ub})T_i - C_i + T_i) \quad (7)$$

EDF の場合,タスクはより早いデッドラインを持つタスクにのみ干渉される.したがってτ_kとτ_iのデッドラインが同じ場合に仕事量が最大となる.[7]さらに EDF でデッドラインミスが起きない場合τ_iの干渉してくる範囲は最大でもデッドライン分(本研究ではT_kと同じ)長さでありその具体例を図 5 に示す.

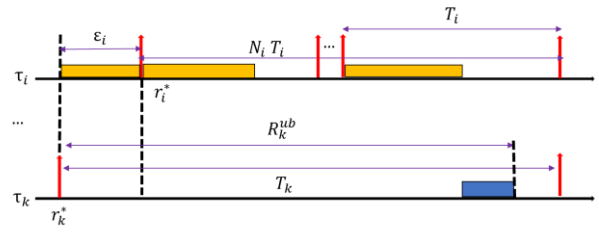

図 5 EDF における仕事量

区間$[r_k^*, r_k^* + T_k)$でのτ_iの仕事量は図 5 より最悪実行時間分がすべて実行できるジョブの実行時間分とリリース時刻が区間$[r_k^*, r_k^* + T_k)$より前でデッドラインが区間内にあるようなジョブ(以下 carry-in ジョブ)の実行時間(図 5 中のε_i部分)を足すことで求めることができる.

最悪実行時間分をすべて実行できるτ_iのジョブの数を$N_i(T_k)$とすると以下の式で求めることができる.

$$N_i(T_k) = \left\lfloor \frac{T_k}{T_i} \right\rfloor \quad (8)$$

また carry-in ジョブの仕事量は以下の式で求めることができる.

$$\varepsilon_i = min(C_i, T_k - N_i(T_k)T_i) \quad (9)$$

よってτ_iの仕事量の最大値は以下の式で求めることができる.

$$W_i^{ub}(T_k) = N_i(T_k)C_i + min(C_i, T_k - N_i(T_k)T_i) \quad (10)$$

ここで補題 4 よりの干渉長の上限値はとなり, 以上から EDF におけるタスクの反応時間の上限値が求められる.

定理 1 EDF でスケジュールされたマルチプロセッサシステムのタスクの反応時間の上限値は,以下の式(11)のR_k^{ub}に関してR_k^{ub}=C_kから始まる不動点反復法で求めることができる.

$$R_k^{ub} \leftarrow C_k + \left\lfloor \frac{1}{M} \Sigma_{i \neq k} \hat{I}_k^i(R_k^{ub}) \right\rfloor \quad (11)$$

ただし,

$$\hat{I}_k^i(R_k^{ub}) = \min\left(\omega_i\left(R_k^{ub}\right), W_i^{ub}(T_k), R_k^{ub} - C_k + 1\right) \quad (12)$$

であり, $W_i^{ub}(T_k)$は式(10)で求められる値である.

Proof.背理法により証明する. 収束したR_k^{ub}の値よりτ_kの値が大きいと仮定する. 収束したR_k^{ub}の値は以下の式(13)で求められる.

$$R_k^{ub} = C_k + \left\lfloor \frac{1}{M} \Sigma_{i \neq k} \min\left(\omega_i\left(R_k^{ub}\right), W_i^{ub}(T_k), R_k^{ub} - C_k + 1\right) \right\rfloor \quad (13)$$

補題 3 より$W_i^{ub} \geq I_k^i\left(r_k^*, r_k^* + R_k^{ub}\right)$が成り立つ.したがって

$$R_k^{ub} \geq C_k + \left\lfloor \frac{1}{M} \Sigma_{i \neq k} \min\left(I_k^i\left(R_k^{ub}\right), R_k^{ub} - C_k + 1\right) \right\rfloor \quad (14)$$

仮定と補題 4 の逆から

$$R_k^{ub} \geq C_k + \left\lfloor \frac{1}{M} M\left(R_k^{ub} - C_k + 1\right) \right\rfloor = R_k^{ub} + 1 \quad (15)$$

この式は矛盾しており. よって背理法より定理が示された.

以上により EDF における反応時間の最大値を計算できるが, 干渉するタスクτ_iの余裕時間の下限値$L_i^{lb} = T_i - R_i^{ub}$を考慮することにより干渉長を低く見積もることが可能である.図 6 は余裕時間の下限値を考慮した状態の図である.

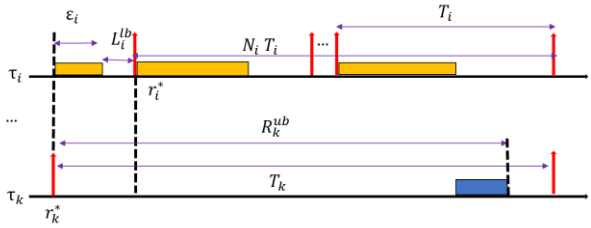

図 6 余裕時間の下限値を考慮した仕事量

したがって図 6 より式(10)は式(16)のように書き換えることができる.

$$W_i^{ub}(T_k) = N_i(T_k)C_i + min(C_i, T_k - N_i(T_k)T_i - L_i^{lb}) \quad (16)$$

ただし, $N_i(T_k)$の値は式(8)の変わらず, $L_i^{lb} < 0$であった場合は$L_i^{lb} = 0$として計算するものとする.

定理 2 タスクセット$\tau = \{\tau_1, \tau_2, \ldots, \tau_N\}$はすべてのタスクが式(17)を満たせば EDF によってスケジュール可能である.

$$L_k^{lb} = T_k - R_k^{ub} \geq 0 \quad (17)$$

EDZL での仕事量の求め方は EDZL と同じである.

提案手法での仕事量の求め方は, 余裕時間がクリティカルなタスクが存在するかで動作が切り替わるが, EDF と EDZL で仕事量が同じため提案手法でも同じ求め方になる.

プロセッサ数を 2 とし, システム利用率が 5%刻みで 50%から 100%までのタスクセットをそれぞれ 1000 個ずつ作成したときの反応時間解析での成功率を図 7 に示す.

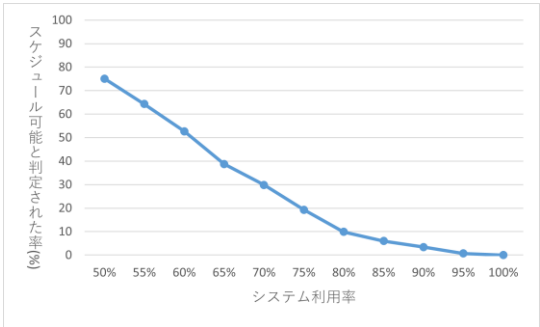

図 7 反応時間解析での成功率

Copyright © 2024 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

3.2 評価

プロセッサ数を 2 とし，システム利用率が 5%刻みで 50%から 100%までのタスクセットをそれぞれ 1000 個ずつ作成しそれぞれのアルゴリズムで処理したときのスケジュール成功率とデッドラインオーバー量をそれぞれ図 8 と図 9 に示す.

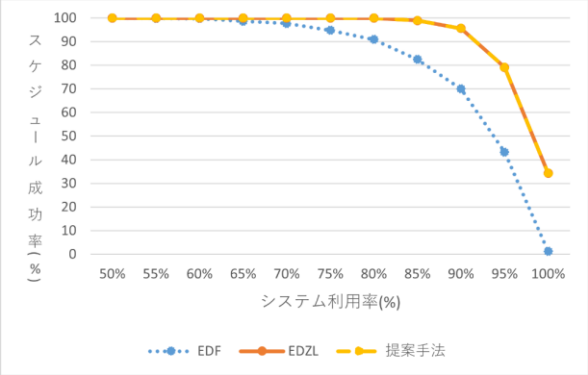

図 8 スケジュール成功率

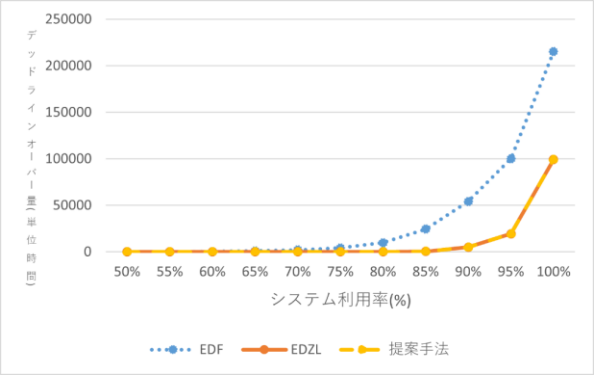

図 9 デッドラインオーバー量

図 8 と図 9 の結果より,提案手法のスケジュール成功率とデッドラインオーバー量は EDZL とまったく同じだということがわかる.また図 10 にスケジューラの起動回数を示す.

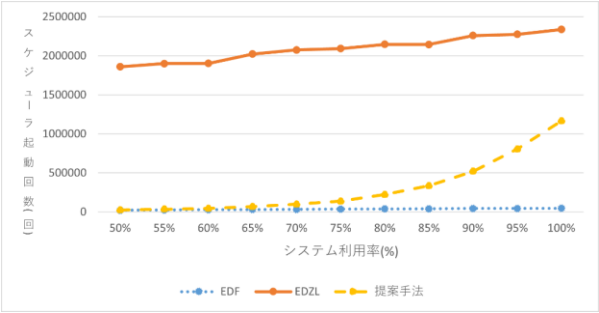

図 10 スケジューラ起動回数

図 10 の結果より,提案手法のスケジューラ起動回数は EDF よりは多いが EDZL と比較して低負荷時は特に削減に成功しているといえる.

4. おわりに

本研究で EDF と EDZL の動作を動的に切り替えることによってスケジュールの成功率を維持したままスケジューラの起動回数を削減することに成功した.しかしこれはシミュレーションでの結果であり,実際の組込みシステムなどに搭載した場合のオーバーヘッドなどは未知数であるため今後の課題とする.

謝辞

本研究は JSPS 科研費 20K11755 の助成を受けたものです

参考文献

[1] 武田瑛, 船岡健司, 加藤真平, 山崎信行, “マルチプロセッサにおけるグローバル RM に基づくリアルタイムスケジューリングアルゴリズム”, 電子情報通信学会技術研究報告, CPSY, 107 巻, 558 号, pp.191-196, (2008)

[2] C. L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a HardReal-Time Environment”, Journal of ACM, Vol. 20, No. 1, pp. 46-67 (1973)

[3] B. Andersson and E. Tovar, “Multiprocessor Scheduling with Few Preemptions”, IP PHurray, Techunical Report, August (2006)

[4] H. Cho, B. Ravindran and E. D. Jensen, “An Optimal Real-Time Scheduling Algorithm for Multiprocessors”, Proceedings of the 27th IEEE Real-Time Systems Symposium, pp. 101-110 (2006)

[5] S. Cho, S. Lee, A. Han, and K. Lin, “Efficient real-time scheduling algorithms for multiprocessor systems”, IEICE Trans Communications, Vol. E85-B, No. 12, pp. 2859-2867 (2002)

[6]M. Bertogna and M. Cirinei: Response-time Analysis for globally scheduled symmetric multiprocessor platforms, 28th IEEE International Real-Time Systems Symposium pp.149-160, 2007

[7] M. Bertogna, M. Cirinei and G. Lipari : Improved schedulability analysis of EDF on multiprocessor platforms, In Proceedings of the EuroMicro Conference on Real-Time Systems, pp209-218, 2005

Copyright © 2024 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.