Vol.2011-SLDM-149 No.16
Vol.2011-EMB-20 No.16
2011/3/18

ハードリアルタイム処理向けマルチコアタスク配置の評価関数設計

鈴木紀章[†]　森義和[††]　岩熊憲二[†††]　枝廣正人[†]

本稿では．ハードリアルタイム制約下におけるマルチコアタスク配置の一手法を提案する．本手法は各タスクの Worst Case Response Time の累積値を評価関数に用いることによって，制御系アプリケーションのタスクセットのデュアルコア配置問題における，コア間依存，待ち時間率，不均衡率の各項目でバランスの取れた配置が可能となった．本手法を，自動車制御系の実システムに適用した結果，タスク数 318 からなるタスクセットの依存元タスク終了待ち時間率を全体の実行時間の 4.1%に抑える配置を実現した．

Designing Evaluation Functions of Multi Core Task Mapping for Hard Real-Time Systems

Noriaki Suzuki[†], Yoshikazu Mori[††], Kenji Iwakuma[†††] and Masato Edahiro[†]

In this paper, we propose a task mapping method for multi-core processors with hard real-time constraints. In this method, a task mapping problems for control systems implemented in a dual core processor that can be balanced with task dependencies across cores, a waiting time rate and a disproportionate rate was achieved, by introducing an accumulation value of the worst case response time of each task for an evaluation function. Experimental result shows that the proposed mapping method needs only 4.1% of the waiting time rate for dependent tasks for the real car control system with 318 tasks.

1. はじめに

制御系アプリケーションでは，その制御対象機器の周期動作の実時間に対応した処理が必要であることから，ハードリアルタイム制約を満たせることが重要である．シングルコア上でハードリアルタイム制約を満たすスケジューリング手法として，Rate-Monotonic(RM)スケジューリング，Earliest-Deadline-First(EDF)スケジューリングが知られている．固定優先度タスクのスケジューリングでは，RM スケジューリングが最適であることが知られている[1][2]．また，動的優先度を用いる場合には EDF スケジューリングが最適である．RM スケジューリングは，固定優先度であるため，そのスケジューリング処理の容易さから多くの RTOS 上で実現可能であり，多数の関連研究も存在する[3][4][5].

一方，マルチコア上で制御系アプリケーションを動作させるには，各周期タスクを動作させるコアを決めるタスク配置が必要である．このタスク配置を静的に決定するシステムにおいて，一般的に周期タスクをマルチコア上に静的に配置する問題は NP-hard 問題であることが知られている[6]．このため，ビンパッキング問題の近似解法を応用したヒューリスティック手法が提案されており, RMFF[7], RMNF, FFDUF[8], RMBF[9]等として知られている．しかし，これらの研究はビンパッキング問題をベースとしているため，なるべく多くのタスクを詰め込むことを指標にしており，コア間依存による待ち時間や，コアの利用率のバランス等の面で課題があった．このため本稿では，RM スケジューリングをマルチコア上で実現する場合の静的タスク配置問題について，配置アルゴリズムが配置の良さを判断する評価関数の一手法について提案する．

本稿では，ハードリアルタイム処理向けマルチコアタスク配置の評価関数を導入し，制御系アプリケーションを想定したタスクセットをデュアルコアに配置する問題に適用した結果について述べる．以下 2 章では，コア間依存，負荷バランス，Worst Case Response Time に基づいた 3 種の評価関数を提案する．3 章では提案する評価関数を用いた配置実験を行い, 各タスクの Worst Case Response Time の累積値を評価関数に用いる手法により，コア間依存，待ち時間率，不均衡率の各項目でバランスの取れた配置が得られること，自動車制御系の実システムのタスクセットを用いた評価で待ち時間率が 4.1%と小さい実用的な配置が実現できたことを示す. 最後に 4 章で本稿をまとめる．

[†] 日本電気株式会社
NEC Corporation
[††] 株式会社 NEC 情報システムズ
NEC Informatec Systems, Ltd.
[†††] 株式会社 日本システムアプリケーション
Japan System Applications Co.,Ltd.

Vol.2011-SLDM-149 No.16
Vol.2011-EMB-20 No.16
2011/3/18

2. 評価関数の設計

本研究では，評価関数として，コア間の負荷バランスの均等化，コア間依存の最小化，各タスクの WCRT(Worst Case Response Time: 最悪応答時間)の累積値の最小化の 3 つを設計した．

2.1 負荷バランス

各コアのコア利用率が均等になることを目標にした評価関数．n 個のタスクからなるタスクセットにおいて，タスク i (i = 0〜n-1)のコア利用率 u_i を，周期 Ti，最悪実行時間 Ci としたときに，$u_i = C_i / T_i$ と定義する．タスク i がコア k に配置されたとき，コア j におけるタスク i のコア利用率 u_{ij} を

$$u_{ij} = \begin{cases} u_i (j = k) \\ 0 (j \neq k) \end{cases}$$

と定義する．この時，コア j のコア利用率 Uj は，

$$U_j = \sum_{i=0}^{n-1} u_{ij}$$

となる．この時，負荷バランスに基づく評価関数 Fu を

$$F_u = \left| U_0 - U_1 \right|$$

と定義し，Fu を最小化するようアルゴリズムを設計する．

2.2 コア間依存

コア間の依存数を評価関数として用いる．コア間に跨る依存の数が n のとき，コア間依存に基づく評価関数 Fd は

$$F_d = n$$

である．ただし，依存数が同一の場合は負荷バランスがより良いものを配置として用いることとする．

2.3 WCRT の累積値

リアルタイムシステムにおいては，デッドラインからの余裕度を大きくすることがシステムの安定性の観点から重要なことがある．ここであるタスクのジョブがリリースされてから，そのジョブの実行が完了するまでの応答時間(Response Time)の最悪値を WCRT と呼ぶが，WCRT が極力が短くなるよう配置すれば．この目的に合致することが想定される．

ここで、シングルコアのもとでの WCRT の算出方法の研究はなされているが[10]，マルチコアの場合はその計算が著しく複雑である．他方，評価関数として用いるうえでの WCRT 計算の厳密性は必須ではなく，配置アルゴリズムから多数回呼び出されることから簡易な計算であることが望ましい．このため，評価関数設計の際，WCRT 計算を行うタスクの直前の依存タスクについては別コアに配置された場合を考慮して計算するが，それ以上の依存を遡った計算については必ずしも行わない手法とした．

つまり n 個のタスクからなるタスクセットにおいて，タスク i (i = 1〜n-1)の最悪実行時間を Ci，シングルコアでの計算方法におけるタスク i の WCRT を Wsi，タスク i が依存するタスク k の WCRT を Wk とするとき，タスク i の WCRT である Wi を

$$W_i = MAX\{Wsi, Wk + Ci\}$$

と定義し，WCRT に基づく評価関数 Fw は，

$$F_w = \sum_{i=0}^{n-1} W_i$$

とする．

3. 配置結果の評価

本研究では，ランダムに生成したタスクセットを用いた場合と，自動車制御系の実システムのタスクセットを用いた場合について評価を行った．評価は，コア間依存数，待ち時間率，不均衡率の 3 つの観点で行った．

待ち時間率は，依存待ち時間を含む全体の実行時間のうち，依存待ち時間の比率を表す．コア間依存数は単純にコア間を跨る依存の数である．コア間依存の評価関数 fd の考え方と等しい．不均衡率は，コア 0 とコア 1 の実行終了時間にどれだけ差が生じているかを表したものである．コア 0 での実行終了時間を Te0，コア 1 を Te1 とした場合，不均衡率 UBr

$$UBr = \frac{\left| Te0 - Te1 \right|}{MAX\{Te0, Te1\}}$$

である．このように不均衡率は，片方のコアに全てのタスクが偏った状態で 100%となり，双方のコアの実行終了時間が全く同一となる場合に 0 となる．

次に，本評価におけるタスクセットの前提条件は次のとおりである．

各タスクの周期は，全タスクで共通とした．優先度は，タスク番号が若い順から高

Vol.2011-SLDM-149 No.16
Vol.2011-EMB-20 No.16
2011/3/18

い優先度を付け，同一優先度のものは存在しないものとした．これは，同一優先度のタスクが存在した場合には，それらのタスクが同時に実行可能となった場合にどちらが先に実行されるか不確定で，評価に用いるのが困難になるためである．今回は，評価関数の検証に的を絞っているため，簡易になるよう制限した．また，タスクスイッチのオーバヘッド，コア間でタスクを起動するオーバヘッドは考えないものとした．更に，タスク間の依存について，必ず優先度が高いタスクから優先度が低いタスクへの依存関係があるものとし，優先度逆転は起きない前提とした．

3.1 ランダムに生成したタスクセット

タスク数が 20 個のタスクセットをランダム生成して，評価を行った．タスクの実行時間 C は，1～10 の均等分布でランダム生成した．周期 T は全タスク同一で，200 固定とした．また，依存がないものについては同一時間に実行可能になる条件とした．

これらのパラメータは，全てのタスクが片方のコアに偏った場合でもデッドラインミスとならない設定である．一般的な評価関数の設計では，デッドラインミスがある場合には評価関数のペナルティ値を加算することでデッドラインミスがない配置を得るよう設計することが考えられる．一方で本研究は，純粋に評価関数の良否を検証することを目的としているため，このような設定を行った．

これらの前提条件のもと，タスク間依存数が 10 個，20 個，30 個，40 個(タスク数の 50%，100%，150%，200%)のタスクセットを，それぞれ 20 個ずつランダム生成し，各評価関数を用いた配置結果の評価を行った．ここで，20 個のタスクに存在し得る最大の依存数は，$_{20}C_2$ =190 個であることから，依存関係の密度は必ずしも高くない．

また，本研究は評価関数の良否を検証することを目的としており，アルゴリズムの高速化等は検討対象外であるため，配置アルゴリズムは全探索とした．

(1) コア間依存数の評価

コア間依存に関する評価結果を，図 1，図 2 及び図 3 に示す．各図において，縦軸が依存数(本)，横軸がタスクセットのタスク間依存数である．図 1 は，各依存数における 20 個のタスクセットの配置結果のうちの最大値，図 2 は最小値を，図 3 は単純平均を示す．また，以後図中及び文中において，負荷バランス，コア間依存及び WCRT に基づく評価関数を「負荷バランス Fu」，「コア間依存 Fd」，「WCRT Fw」と表記する．

図に示すように，「負荷バランス Fu」と「WCRT Fw」を比較した場合，いずれの依存数においても最大，最小，平均全てで「WCRT Fw」が良好な結果を得ていることがわかる．一方で「コア間依存 Fd」の場合は，コア間依存が 0 となる(依存待ちが発生しない)配置のみ得られるため待ち時間は発生しない結果となっているが，その問題点については後述する．

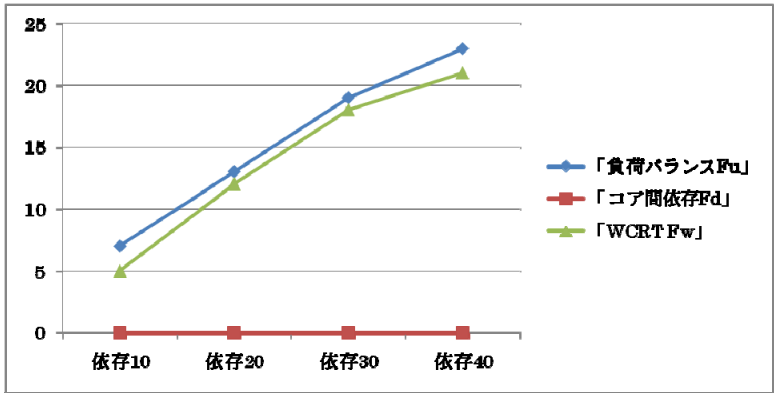

図 1　コア間依存(最大)

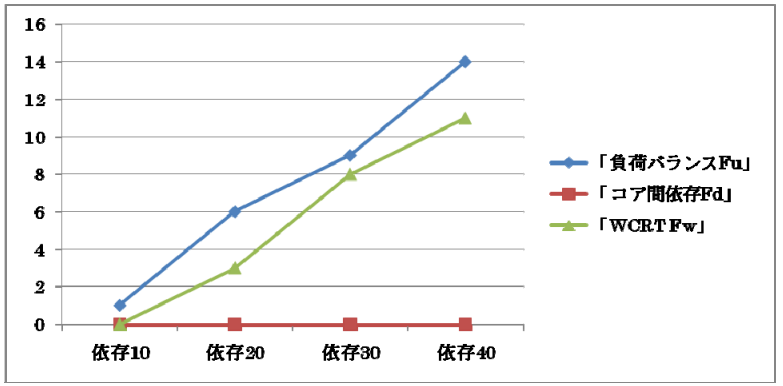

図 2　コア間依存(最小)

Vol.2011-SLDM-149 No.16
Vol.2011-EMB-20 No.16
2011/3/18

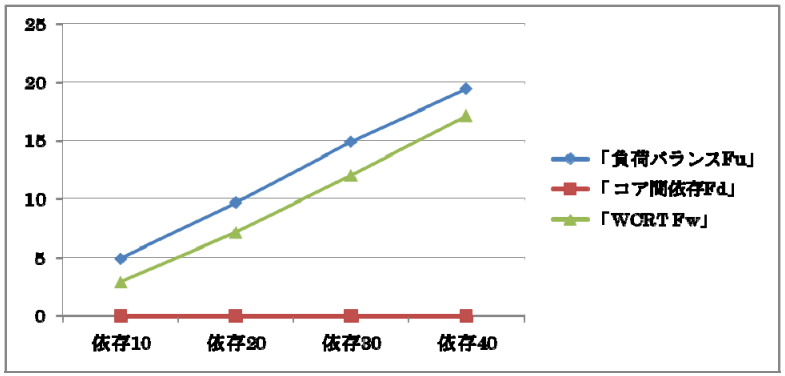

図 3　コア間依存(平均)

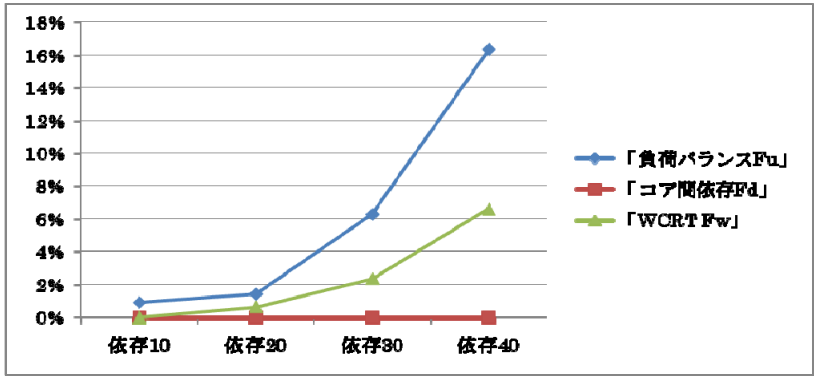

図 5　待ち時間率(平均)

(2) 待ち時間率の評価

待ち時間率に関する評価結果を，図 4 及び図 5 に示す．各図において，縦軸が待ち時間率(%)，横軸がタスクセットのタスク間依存数である．図 4 は，各依存数における 20 個のタスクセットの配置結果のうちの最大値を，図 5 は単純平均を示す．ここで，最小の待ち時間率はいずれの試行でも 0 であったため，図示しない．

図に示すように，「負荷バランス Fu」と「WCRT Fw」を比較した場合，いずれの依存数においても最大，平均双方で「WCRT Fw」が良好な結果を得ていることがわかる．特に平均を見た場合，「負荷バランス Fu」は「WCRT Fw」の 2 倍以上の依存待ち時間

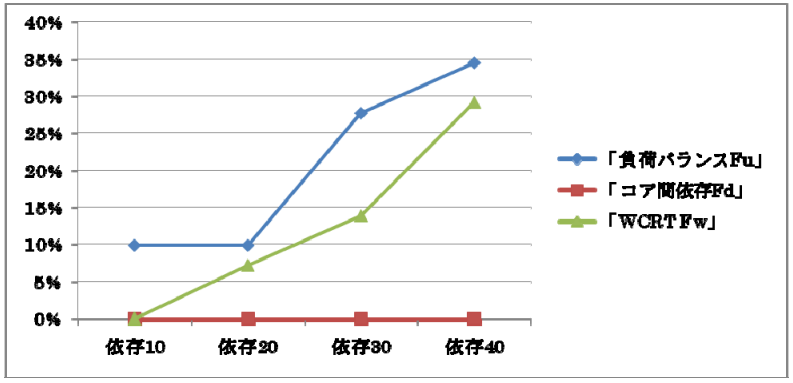

図 4　待ち時間率(最大)

が発生し，依存数 40 の場合は 16%に達している．このとき，「WCRT Fw」は 6.5%ほどである．「コア間依存 Fd」の場合は，コア間依存が 0 となる(依存待ちが発生しない)配置のみ得られるため待ち時間は発生しない結果となっているが，その問題点については後述する．

(3) 不均衡率の評価

不均衡率に関する評価結果を，図 6，図 7 及び図 8 に示す．各図において，縦軸が不均衡率(%)，横軸がタスクセットのタスク間依存数である．図 6 は，各依存数における 20 個のタスクセットの配置結果のうちの最大値，図 7 は最小値を，図 8 は単純平均を示す．

図に示すように，「負荷バランス Fu」と「WCRT Fw」を比較した場合，コア間の依存数が少ない場合に「WCRT Fw」の不均衡率がやや大きい結果となっているが，依存数が増加するに従って差が小さくなっていることがわかる．「負荷バランス Fu」では，依存数が少ない，つまりコア間依存の存在に伴う依存待ち時間が発生しにくい状況では，負荷を均等にすることで終了時刻等になる傾向があるが，依存数の増加に伴って依存待ち時間の影響が大きくなっていくことがわかる．一方で「WCRT Fw」では，依存数が少ない場合には必ずしも終了時刻を均等にする方向には効果が出ていないが，その評価関数値計算にコア間依存の影響が盛り込まれていることから，依存数が増加するにしたがってその効果が表れてきている状況が確認できる．

ここで，コア間依存数と待ち時間率で良好な結果に見えていた「コア間依存 Fd」の場合について見てみると，実際には不均衡率が非常に大きくなり，2 つのコアを効果的に利用できていないことがわかる．特に，最大では依存数 10 の場合でも 80%以上，

Vol.2011-SLDM-149 No.16
Vol.2011-EMB-20 No.16
2011/3/18

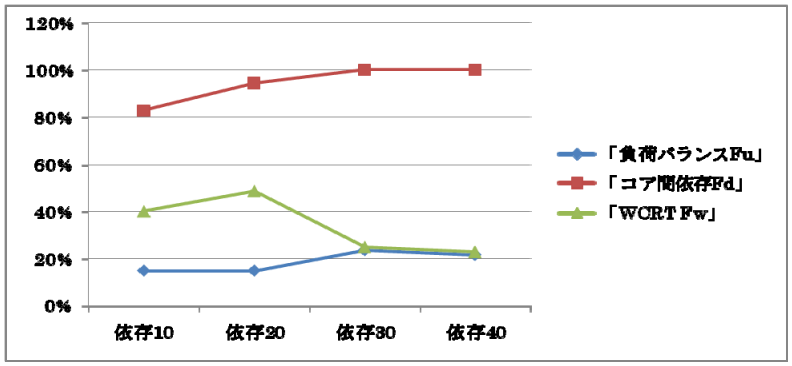

図 6　不均衡率(最大)

依存数 30 以上では 100%，つまり完全に片方のコアに全タスクが存在する状態が散見され，デュアルコアのメリットが得られていない．

このように，コア間依存を最小化する「コア間依存 Fd」の手法では，タスクセットによっては理想的な配置とはかけ離れた結果になる可能性があることが確認できる．一例として，図 9 のようなタスクセットが挙げられる．ここに示すように，タスク A のみが独立しており，他のタスクには何らかの依存関係がある場合，コア間依存を最小化する「コア間依存 Fd」の手法では片方のコアに負荷が集中することがわかる．一方で，理想的な配置の一例は，コア間の依存が 1 つ残る場合であり，必ずしも評価関数値が配置の良さを表していないことが分かる．このため，「コア間依存 Fd」を評価関数に用いる場合には負荷バランスが均等な条件内で探索を行うなど，アルゴリズムの配慮が必要である．

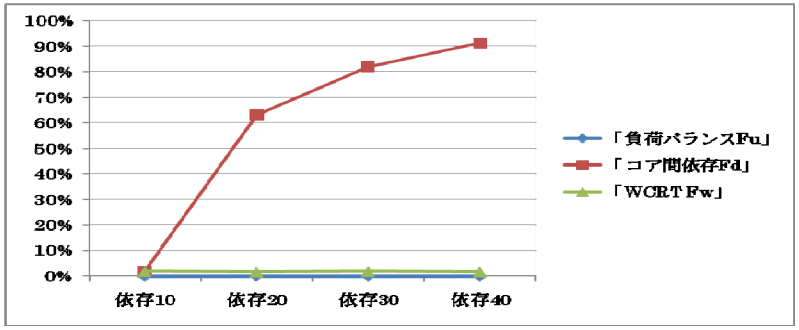

図 7　不均衡率(最小)

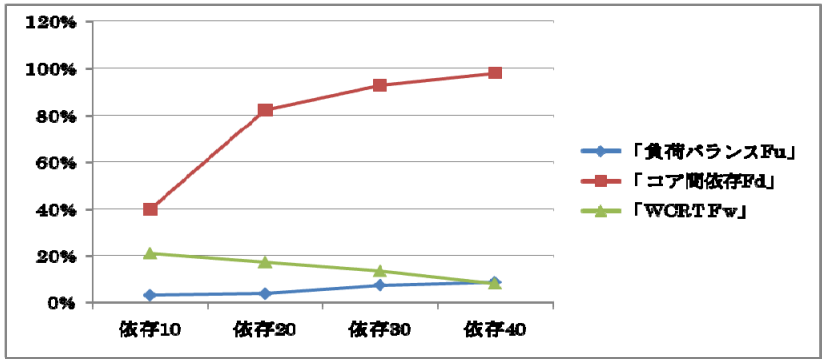

図 8　不均衡率(平均)

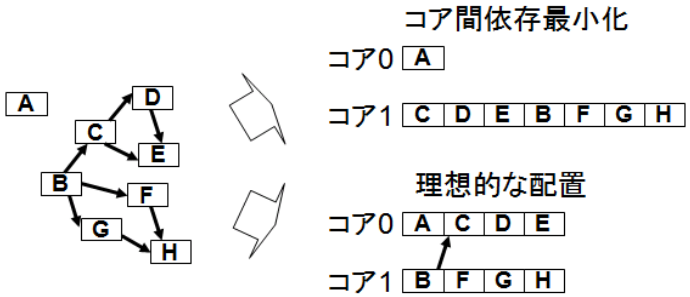

図 9　コア間依存を用いた場合の失敗例

3.2　実システムのタスクセットの例

ここでは，制御系の実システムで用いられているタスクセットを用い，実験を行った結果について示す．タスクセットは，株式会社本田技術研究所のご協力により，自動車制御用アプリケーションのタスクセット情報をご提供頂いた．

タスクセットは，タスク数 318，依存数 773 の構成である．タスクの実行時間は，最小 3，最大 195 である．最短周期は 10000 で，全タスクの実行時間合計は 7425 となっている．

本タスクセットでは，タスク数が多いため，現状では最適解の厳密解を現実的な時間内に求めることができない．このため，今回はグリーディー法をベースにした近似解法を用い，配置後に手動で外乱を与えて再度配置を繰り返す作業により近似解を求めた．従って，この結果よりも評価関数値が良くなる配置が存在し得る．

Vol.2011-SLDM-149 No.16
Vol.2011-EMB-20 No.16
2011/3/18

このような実システムのタスクセットを用い，「WCRT Fw」の評価関数を用いて配置した結果を，表 1 に示す．

表 1　実システムのタスクセットの配置結果

項目	結果
コア間依存	22
待ち時間率	4.2%
不均衡率	15.1%

表 1 に示すように，コア間依存は 22 個と，タスク数及び依存数に対して十分小さい配置が得られた．また，不均衡率は 15.1%とややあるものの，待ち時間率は 4.1%と小さくなっており，実用的な配置を得ることができた．

3.3 評価のまとめ

タスク数が 20 個のタスクセットをランダム生成して，「負荷バランス Fu」，「コア間依存 Fd」，「WCRT Fw」の各評価関数を用いた配置結果の評価を行った．コア間依存数及び待ち時間率の評価において，「WCRT Fw」は「負荷バランス Fu」を上回る結果となった．一方で「コア間依存 Fd」はコア間依存数及び待ち時間率で最も優秀であるものの，不均衡率の評価により片方のコアにタスクが偏る弊害が明らかとなった．また，「WCRT Fw」は不均衡率でもコア間依存数が多い場合に「負荷バランス Fu」に匹敵する性能が得られた．このような結果から，各タスクの Worst Case Response Time の累積値を評価関数に用いる手法により，コア間依存，待ち時間率，不均衡率の各項目でバランスの取れた配置が得られることが明らかとなった．

また、自動車制御系の実システムのタスクセットを用い，「WCRT Fw」の評価関数を用いて最適タスク配置の近似解を得た結果，不均衡率は 15.1%とややあるものの，待ち時間率が 4.1%と小さい実用的な配置が実現できた．

4. おわりに

本稿では，ハードリアルタイム処理向けマルチコアタスク配置の評価関数に基づいた配置手法を提案した．最適な配置方法を得るために，コア間の負荷バランス，コア間の依存, 各タスクの WCRT の累積値の 3 種類の評価関数を使った配置手法に対して，コア間依存数，待ち時間率，不均衡率の 3 つの検証項目について比較検討を行った．まず、ランダムに生成したタスクセットを使った評価では，各タスクの WCRT の累積値を評価関数に用いた場合に，各評価項目でバランスの取れた配置を得られた．一方自動車制御系の実システムのタスクセットを用いた評価では，WCRT の累積値を評価関数に用いた配置手法により，コア間依存は 22 個と，タスク数及び依存数に対して十分小さい配置が得られた．また，不均衡率は 15.1%とややあるものの，待ち時間率は 4.1%と小さくなっており，実用的な配置を得ることができた．

謝辞　本研究を進めるにあたって，実例として自動車制御系アプリケーションのタスクセット情報をご提供頂き，検討にご助力頂いた，株式会社本田技術研究所の，荒木秀和氏，大口良輔氏，岩城喜久氏に，謹んで感謝の意を表する．

参考文献

1)　C. L. Liu and J. Layland: Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, J. Assoc. Comput. Machinery 10(1), pp.174-189 (1973).
2)　P. Serlin: Scheduling of Time Critical Processes, Proceedings of the Spring Joint Computers Conference 40, pp.925-932 (1972).
3)　J.W.S. Liu, K.-J. Lin and S. Natarajan: Scheduling Real-time, Periodic Jobs Using Imprecise Results, IEEE Real-Time Systems Symposium, pp.252-260 (1987).
4)　S. Ramos-Thuel and J.K. Strosnider: The Transient Server Approach to Scheduling Time-Critical Recovery Operations, IEEE Real-Time Systems Symposium, pp.286-295 (1991).
5)　W-K. Shin, J.W.S. Liu, AND J-Y Chung: Fast Algorithms for Scheduling Imprecise Computations: IEEE Real-Time Systems Symposium, pp.12-19 (1989).
6)　J.Y.T. Leung and J. Whitehead: On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks, Performance Evaluation 2, pp237-250 (1982).
7)　S. K. Dhall and C. L. Liu: On a Real-Time Scheduling Problem, Operations Research , vol. 26, no. 1, pp.127-140(1978).
8)　S. Davari and S.K. Dhall: An on line algorithm for rea-time allocation, 19thAnn. Huwuii Int'l Conf: System Sciences, pp. 133-141(1986).
9)　Yingfeng Oh and Sang H. Son: Tight Performance Bounds of Heuristics for a Real-Time Scheduling Problem, Technical Report, CS-93-24, University of Virginia Charlottesville, VA, USA(1993).
10)　M. Joseph and P. Pandya : Finding Response Times in a Real-Time System, The Computer Journal, vol　29, no.5, pp.390-395(1986).