性能が不均等なマルチプロセッサにおける タスク間の実行公平性を実現するロードバランス機構

石田 利永子[†] 本田 晋也[†] 高田 広章[†]
福井 昭也[††] 鈴木 均[†††]
安達 浩次[†††] 田原 康宏[†††]

組込みシステムの分野でもマルチプロセッサが使用されるようになってきた．さらに，プロセッサ処理能力の向上により，マルチストリーミング処理が可能となりつつある．そこで，我々は組込み向けマルチプロセッサ用 RTOS 上で動作する，マルチストリーミング処理を対象としたロードバランス機構である均等型仕事量平準化方式を提案した．この提案は性能が均等なマルチプロセッサ環境を想定している．しかし，ハードリアルタイムタスクが共存する場合など，マルチストリーミング処理が使用することができるプロセッサ性能が不均等となるマルチプロセッサ環境上でもロードバランスが必要な場合が想定される．そこで，本研究では性能が不均等なマルチプロセッサ環境上でも，マルチストリーミング処理の時間制約を満たすことができるロードバランス機構を提案する．提案手法をマルチプロセッサ RTOS 上で実現し，有効であることを確認した．

Load balance mechanism realizing fair progress among tasks in multiprocessor with asymmetric performance

Rieko Ishida,[†] Shinya Honda,[†] Hiroaki Takada,[†] Akiya Fukui,[††]
Hitoshi Suzuki,[†††] Koji Adachi[†††] and Yasuhiro Tawara[†††]

Our workload leveling mechanism on the RTOS for an embedded multiprocessor has been enhanced from that for a symmetric-performance multiprocessor into that for an asymmetric-performance multiprocessor(APM). On the RTOS for embedded multiprocessor, hard realtime tasks can be stuck asymmetrically to the processors with higher priority than soft realtime tasks. That makes an APM from the standing point of soft realtime tasks. Using multi-streaming applications as soft realtime tasks, we demonstrated effectiveness of our workload leveling mechanism for an APM.

1. はじめに

近年，汎用計算機のみならず組込みシステムの分野においても，マルチプロセッサの重要性が急速に増している．リアルタイム性が求められる組込みシステムでは，予測可能性や検証性の観点から，タスクを各プロセッサに静的に配置する方法が一般的である[1)2)]．しかしながら，システムによっては，ハードリアルタイムタスクだけではなく，要求される時間制約が弱いソフトリアルタイムタスクを含む場合がある[3)]．このようなシステムの場合，ハードリアルタイムタスクはリアルタイム性を確保するためにプロセッサに静的に割り付け，ソフトリアルタイムタスクはスループットの向上のためにロードバランスを行い，動的にプロセッサに割り付ける方法が考えられる．

ソフトリアルタイムタスクを含むアプリケーション（以下，ソフトリアルタイムアプリケーション）として代表的なものに，マルチストリーミング処理がある．マルチストリーミング処理とは，複数の動画や音声などのストリームデータを同時にエンコード，デコードする処理である．マルチストリーミング処理の入力データとなるストリームデータは，動画の場合はフレームレート，音声の場合は処理単位（以下，ブロック）の大きさが異なるので，単位時間分のストリームデータを処理するために必要なプロセッサ時間は異なる．単位時間分のストリームデータを処理するために必要な

† 名古屋大学大学院情報科学研究科
Graduate School of Information Science, Nagoya University
†† 株式会社ルネサスソリューションズ
Renesas Solutions Corp.
††† ルネサスエレクトロニクス株式会社
Renesas Electronics Corporation.

プロセッサ時間が長いことを，負荷が高いと定義する．さらに，動画処理の場合，各フレームの負荷は動的に変動するので処理時間の予測がつかないという特徴を持つ．以上より，マルチストリーミング処理の時間制約を可能な限り満たすためには，動的な負荷の変動に基づいた指標によるロードバランス機構を適用し，マルチプロセッサの能力を有効活用する必要がある．

そこで我々は，マルチストリーミング処理に対するロードバランス機構として，均等型仕事量平準化方式を提案した[4]．本方式は，各タスクのデッドラインまでの時間を平準化するようにロードバランスを行う．そして，デッドラインまでの時間が平準化されている状態を，タスクの実行公平性が実現できていると定義する．実行公平性が実現できていれば，時間制約を満たすことができる可能性が高くなると考えた．しかしながら，この方式は，個々のプロセッサの性能が異なる状況を考慮しておらず，性能が不均等なマルチプロセッサ環境上で適用すると時間制約を満たせない可能性があることがわかった．

本研究では，性能が不均等なマルチプロセッサ環境上でも有効なロードバランス機構として，不均等型仕事量平準化方式を提案する．性能が不均等なマルチプロセッサ環境としては，各プロセッサにハードリアルタイムタスクが共存することにより，ロードバランスの対象アプリケーションが利用することができるプロセッサ時間が不均等になる状況を検討する．提案した方式を RTOS 上に実現して評価し，性能が不均等なマルチプロセッサ環境上で，マルチストリーミング処理を簡略化したアプリケーションに適用した結果，均等型仕事量平準化方式と比較して，実行公平性を保つことができる可能性が高いことを確認した．

本論文の構成は，まず 2 章でロードバランス対象とするマルチストリーミング処理の特徴と均等型仕事量平準化方式ついて述べる．3 章で，性能が不均等なマルチプロセッサ環境上で，均等型仕事量平準化方式を適用した場合に生じる貼りつき問題について述べる．4 章で，不均等型仕事量平準化方式の概要と，貼りつき問題解消方法について述べる．5 章では，性能が不均等なマルチプロセッサ環境上で，不均等型仕事量平準化方式を適用した結果の評価について述べ，6 章で，関連研究について述べる．最後 7 章では，本論文のまとめと今後の展開について述べる．

2. 背　　景

本章では，まず対象とするマルチストリーミング処理の特徴について述べ，次に均等型仕事量平準化方式について述べる．

2.1 マルチストリーミング処理の特徴

本研究の対象とするソフトリアルタイムアプリケーションは，マルチストリーミング処理とする．

エンコード処理の入力となる RAW データ，デコード処理の入力となる圧縮データを，ストリームデータと総称する[5]．マルチストリーミング処理の特徴を次に示す．

(a) ストリームデータの同時処理最大数はプロセッサ数を超える．
(b) ストリームデータごとに，タスクを割当て処理を行う．
(c) ストリームデータごとに，単位時間分のデータを処理するために必要なプロセッサ時間が異なる．
(d) 同一ストリームデータであっても，処理に要するプロセッサ時間は変動し予想不可能である．
(e) エンコード処理の入力となる RAW データは，フレームレートないしブロックの 1 単位ごとに，入力 Buffer に格納される．
(f) デコード処理の出力である再生データは，出力 Buffer に格納され，フレームレートないしブロックの 1 単位ごとに再生される．

エンコード処理では，特徴（e）で述べた RAW データを，入力 Buffer に空き領域があり格納できる状態を時間制約を満たしているとする．デコード処理では，特徴（f）で述べた出力 Buffer に再生データが存在し，再生が途切れない状態を時間制約を満たすとする．

2.2 実行公平性

マルチストリーミング処理は，特徴（d）より負荷の変動は予測不可能であるので，時間制約を完全に満たす方法はないが，ソフトリアルタイム処理であるため，可能な限り時間制約を満たせればよい．

そこで，マルチストリーミング処理における時間制約を満たしやすい状態について述べる．

まず個々のタスクに着目すると，エンコード処理の時間制約を満たすには，常に必要な分の RAW データを受信できるように，入力 Buffer の空き領域を確保する必要がある．一方，デコード処理の時間制約を満たすには，常に出力 Buffer に再生に必要な分の再生データが存在する必要がある．ここで，エンコード処理における入力 Buffer の空き領域に入れることができる RAW データによるデータ時間，および，デコード処理の出力 Buffer の再生データによる再生時間をまとめて余裕時間と定義する．すなわち，あるタスクの余裕時間が 0 となると，そのタスクは時間制約を満たせなくなる．

次に，システム全体に着目すると，各タスクの余裕時間が平準化されていると，時間制約を満たせる可能性が高くなると考えられる．

2.3 均等型仕事量平準化方式

均等型仕事量平準化方式とは，ロードバランス対象タスクの仕事量を平準化するロードバランス機構である．マルチストリーミング処理に均等型仕事量平準化方式を適用するには，余裕時間を仕事量とする．

ロードバランス機構を実現する OS 内のモジュールもしくはミドルウェアをロードバランスモジュールと呼び，プロセッサ内ロードバランスモジュールと，プロセッサ間ロードバランスモジュールに大別し，それぞれが独立して動作する．

2.3.1 プロセッサ内ロードバランスモジュール

プロセッサ内で仕事量が最小のタスクを最高優先順位にすることで，実行を優先する．このことにより，プロセッサ内のタスクの仕事量は平準化される．この処理はプロセッサごとに独立して行う．

2.3.2 プロセッサ間ロードバランスモジュール

プロセッサごとに割り付けられているタスクの仕事量の平均を，プロセッサの進捗度と定め，定期的に進捗度に応じてタスクをマイグレーションする．

同一プロセッサに割り付けられているタスクの仕事量はプロセッサ内ロードバランスで平準化されている．そこで進捗度の小さいプロセッサから，進捗度が最大のプロセッサへタスクをマイグレーションすることにより，システム全体の仕事量を平準化する．具体的には，プロセッサごとの進捗度の最大値と，判断処理を行っているプロセッサ（以下，自プロセッサ）の進捗度の差がプロセッサ間ロードバランス判断基準値（以下，LB 判断値）以上なら，ロードバランスが必要と判断する．マイグレーション対象となるタスクは，自プロセッサ内で 2 番目に仕事量の小さいタスクとする．

3. 性能が不均等なマルチプロセッサ環境におけるロードバランス

本章では，均等型仕事量平準化方式を，性能が不均等なマルチプロセッサ環境へ適用した場合に発生する問題である，貼りつき問題について述べる．

3.1 性能が不均等なマルチプロセッサ環境

本研究では，ソフトリアルタイムタスク等のロードバランス対象タスク（以下，LB タスク）と，ロードバランスの対象とならないハードリアルタイムタスク（以下，HR タスク）が共存する環境を対象とする．

HR タスクの特徴を下記に示す．

- 動的起動及び終了は行わない

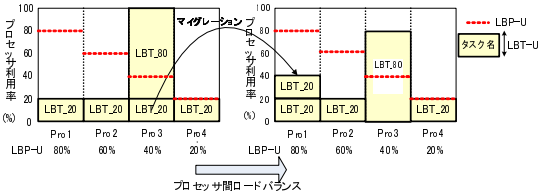

図 1 貼りつき問題
Fig. 1 Stick Problem

- 周期と実行時間は固定であり，設計時に定まる
- 各タスクのプロセッサへの割付けは設計時に決定し，実行時に変更することはない

HR タスクには，LB タスクより高い優先度を与えることにより，LB タスクに優先してプロセッサ時間を割当てる．LB タスクは，HR タスクがプロセッサを使用していない時間を利用して処理を行う．そのため，LB タスクが利用することができるプロセッサ利用率（以下，LBP-U）は異なるため，不均等となる．なお，本研究では，各プロセッサの LBP-U は動的に変動しない．

3.2 均等型仕事量平準化方式適用時の問題

性能が不均等なマルチプロセッサ環境上での均等型仕事量平準化方式の問題点について述べる．

性能が不均等なマルチプロセッサ環境として，4 個のマルチプロセッサの各プロセッサに HR タスクが存在する場合を例にして説明する．プロセッサ 1，2，3，4 の LBP-U はそれぞれ，80%，60%，40%，20%とする．

LB タスクの負荷は，必要プロセッサ利用率（以下，LBT-U）で表現する．LBT-U とは，LB タスクが単位時間分のデータを処理するのに必要な処理時間の，デッドラインまでの時間に対する割合である．

均等型仕事量平準化方式は，各プロセッサの LBP-U が均等であるという前提で設計したアルゴリズムであるので，仕事量が最小の LB タスクをマイグレーション対象としない．その結果，次に述べる 2 点の問題が発生することを貼りつき問題と呼ぶ．

(1) 時間制約を満たせない

図 1 左の状況では，プロセッサ 3 の進捗度は，他プロセッサと比較して小さくなるので，ロードバランスが必要と判断する．均等型仕事量平準化方式では，仕事量が 2 番目に小さい LBT_20 がマイグレーション対象となる．その結果，LBT_80 は LBP-U が 40%であるプロセッサ 3 で稼働を続けるので，時間制約を満たすのに十分なプロセッサ時間が割当てられない（図 1 右）．

(2) 実行公平性が実現できない

プロセッサ 4 に割付けられている LB タスクは 1 タスクのみのため，ロードバランスは必要ないと判断する．プロセッサ 4 に割付けられている LBT_20 は，時間制約を満たすことができるが，プロセッサ 1 やプロセッサ 2 に割付けられている LBT_20 と比較すると割当てられるプロセッサ時間が少ないために，仕事量が増加しない．よって，時間制約は満たしていても実行公平性が実現できない．

4. 不均等型仕事量平準化方式

本章では，3 章で述べた貼りつき問題の解決方法として，不均等型仕事量平準化方式を提案する．

4.1 概　　要

不均等型仕事量平準化方式では，均等型仕事量平準化方式に加えて，貼りつき問題の解消が必要となる．貼りつき問題解消には，プロセッサ間の性能が不均等となる状況を考慮し，貼りつき問題を検知し，解消する仕組みが必要となる．

不均等型仕事量平準化方式を検討する際の前提として，下記 2 点を定めた．

(1) 均等型仕事量平準化方式をベースとして，貼りつき問題発生時に自プロセッサ内の仕事量が最小のタスク（以下，仕事量最小タスク）のマイグレーション方法を検討する．

(2) LBT-U の値はアルゴリズム中で使用しない

(1) は，均等型仕事量平準化方式をベースとすることにより，性能が均等なマルチプロセッサ環境上にも適用可能となる．また，プロセッサ内ロードバランスモジュールはそのまま適用する．(2) は，対象とするマルチストリーミング処理は，特徴（d）より負荷は動的に変動するため定めた．

4.2 貼りつき問題の解決

貼りつき問題の解決として，4.1 節で述べた前提 (1) より，均等型仕事量平準化方式のプロセッサ間ロードバランスモジュールに，貼りつき問題解消モジュールを追加し，各プロセッサで周期的に処理を行う．貼りつき問題解消モジュールでは，貼りつき問題を検知した場合，仕事量最小タスクのマイグレーション先の決定を行い，マイグレーションをする．貼りつき問題が検知されない場合は，均等型仕事量平準化方式のプロセッサ間ロードバランスモジュールを適用する．以上より，不均等型仕事量平準化方式のプロセッサ間ロードバランスモジュールは図 2 となる．

4.2.1 貼りつき問題発生の検知

本研究では，自プロセッサより LBP-U が大きいプロセッサの進捗度の最大値と，仕事量最小タスクの仕事量との差が，LB 判断値以上の場合に貼りつき問題が発生しているとした．

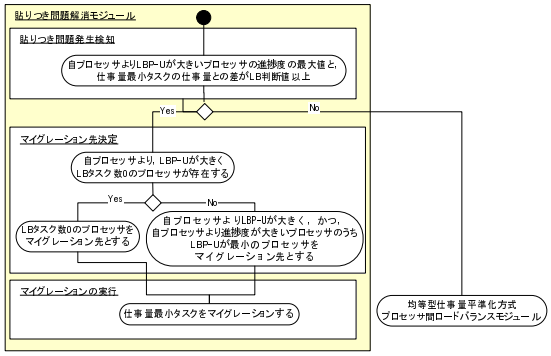

図 2 不均等型仕事量平準化方式のプロセッサ間ロードバランスモジュール

Fig. 2 Load balance among processors of Unequality type Workload leveling mechanism

ここで，自プロセッサより LBP-U が大きい他プロセッサのみ対象とするのは，自プロセッサより LBP-U が小さいプロセッサへマイグレーションしても，十分なプロセッサ時間を割当てることができないので，マイグレーション先の対象としないためである．

4.2.2 マイグレーション先の決定

貼りつき問題を検知すると，マイグレーション先を決定する必要がある．マイグレーション先は，自プロセッサより LBP-U が大きいプロセッサの中から，以下の (1)(2) の順に選択する．

(1) LB タスク数が 0 のプロセッサ

(2) 進捗度と仕事量最小タスクの仕事量の差が LB 判断値以上のプロセッサのうち，LBP-U が最小のプロセッサ

(2) において，自プロセッサより LBP-U が大きいプロセッサのうち進捗度最大のプロセッサとする方法もあるが，複数のプロセッサから進捗度最大のプロセッサに LB タスクのマイグレーションが集中し，その結果さらなるマイグレーションが必要となる可能性を考慮し選択しない．最後に，仕事量最小タスクを，決定したマイグレーション先プロセッサにマイグレーションする．

5. 評　　価

本章では，不均等型仕事量平準化方式を，組込みシステム用マルチプロセッサリアルタイム OS である，TOPPERS/FMP カーネル（以下，FMP カーネル）上に実現して，マルチストリーミング処理のうち，デコード処理を簡略化したプログラムに適用した．

評価環境は，600MHz の 4 プロセッサの SH4A-

MULTI を用いた．HR タスクを各プロセッサに割付けることで，プロセッサ 1，プロセッサ 2，プロセッサ 3，プロセッサ 4 の LBP-U をそれぞれ，80%，60%，40%，20%と不均等にした．

5.1 評 価 項 目

上記の環境で，均等型仕事量平準化方式と比較し，不均等型仕事量平準化方式が，LB タスクの実行公平性を実現できることを確認する．下記 2 点を評価項目とする．

(1) 時間制約を満たしている（失敗率が低い）こと
(2) 仕事量が平準化されていること

各タスクが時間制約を満たせなくなるのは，余裕時間（仕事量）が 0 となったときである．つまり，仕事量が 0 であれば時間制約を満たせずロードバランスが失敗しているとみなす．そこで (1) に関しては，仕事量を 10 ミリ秒ごとに取得し，仕事量が 0 である割合を失敗率として定義し評価することにした．実行公平性を実現するには，少なくとも (1) を満たす必要がある．

次に (2) を評価する．各 LB タスクの仕事量変化をグラフで表し，均等型仕事量平準化方式と比較して，不均等型仕事量平準化方式を適用した場合に，仕事量が平準化されているかどうかを評価する．

5.2 デコード処理タスク

マルチストリーミング処理の特徴（c）に対応するため，LBT-U の異なるタスクを含むタスクセットを用いる．前述したように LBT-U とは，タスクの処理時間のデッドラインまでの時間に対する割合である．例えば，フレームレートが 8fps のストリームデータでは，1 フレーム分の処理を行うと 125 ミリ秒分 (1000/8) の再生データを生成するため，仕事量を 125 加算する．この場合の，デッドラインまでの時間は 125 ミリ秒となる．1 フレーム分の処理時間を 50 ミリ秒とすると，時間制約を満たすのに必要なタスクのプロセッサ利用率は，40%(50/125) である．この値が大きいほど負荷が高いと言える．

タスクの処理はデコード処理を模擬し，可能な限りキャッシュの影響を受けないようにするため，ループ処理とした．ループ処理終了後にフレームレートに応じた仕事量を加算する．

マルチストリーミング処理の特徴 (f) に対応するため，1 ミリ秒周期の周期ハンドラ中で，各タスクの仕事量を 1 ミリ秒ごとに 1 ずつ減少する．

表 1 LB タスクの LBT-U と起動時の割付けプロセッサ
Table 1 LBT-U of LBtasks and Layout processor at the time of the start

タスクセット	プロセッサ 1 ※ 80%	プロセッサ 2 ※ 60%	プロセッサ 3 ※ 40%	プロセッサ 4 ※ 20%	LBT-U 合計	LB 判断値
1	task1_1 15[333]	task2_1 15[333]	task3_1 30[167]	task4_1 50[100] task4_2 50[100]	160	333
2	task1_1 20[250]	task2_1 20[250]	task3_1 20[250]	task4_1 20[250] task4_2 20[250] task4_3 20[250]	120	250
3	task1_1 30[167]	task2_1 30[167]	task3_1 30[167]	task4_1 30[167] task4_2 30[167]	150	167

※ LBP-U を示す．
上段：LB タスク名を示す．
下段：LBT-U(%) に続き，[] 内に 1 フレームごとに加算する仕事量を示す．

表 2 タスクセットごとの失敗率平均（%）
Table 2 Missrate average every task set

	1	2	3
均等型仕事量平準化方式	11.5	0.0	5.3
不均等型仕事量平準化方式	0.1	0.0	0.0

5.3 評 価 条 件

プロセッサ間ロードバランス周期は 50 ミリ秒とした．各タスクが処理するフレーム数は 100 とした．また，1 フレームの処理時間を 50 ミリ秒としたが，フレームレートが異なるので LBT-U が異なる．タスクセットは 3 種類用意した．表 1 に，LB タスクとその LBT-U，起動時の割付けプロセッサ，LB 判断値を示す．全 LB タスクの必要プロセッサ利用率の合計は，LBP-U の合計である 200%を超えない．LB タスクの負荷が偏った割付けであっても，ロードバランスにより仕事量が平準化されることを確認するため，LBP-U が最小のプロセッサ 4 の負荷が高くなるよう，偏った割付けとしている．LB 判断値は，LB タスクの 1 フレームごとに加算する仕事量の最大値とした．

5.4 評価結果と考察

それぞれのタスクセットに対して，均等型仕事量平準化方式と，不均等型仕事量平準化方式を適用した．

まず，評価項目 (1) を評価する．表 2 にタスクセットごとの失敗率の平均を示す．すべてのタスクセットにおいて，不均等型仕事量平準化方式の失敗率平均は 0%に近い値を示し，均等型仕事量平準化方式と比較して，高い優位性を示した．このことより，不均等型仕事量平準化方式は均等型仕事量平準化方式と比較して，時間制約を満たす可能性が高いといえる．

次に評価項目 (2) を評価する．各タスクセットに両方式を適用し，各 LB タスクの割付けプロセッサの遷移と仕事量変化を観察した．

タスクセット 1 の結果を図 3 に示す．均等型仕事量平準化方式では，プロセッサ間ロードバランスによ

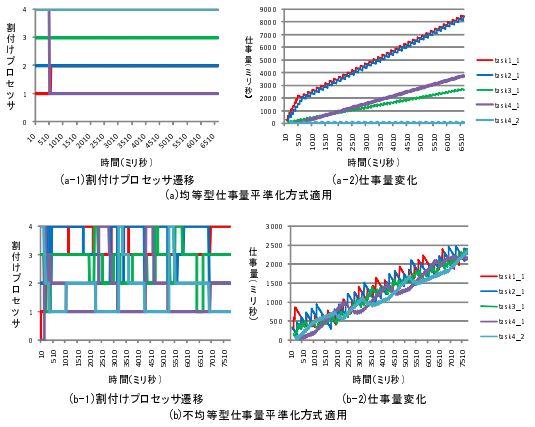

図 3 タスクセット 1
Fig. 3 Taskset1

り，task4_1 がプロセッサ 1 にマイグレーションされ，LBT-U50%の task4_2 は LBP-U20%のプロセッサ 4 で稼働を続け（貼りつき問題）（図 3(a-1)），仕事量を平準化する以前に時間制約を満たすことができない．また，LBT-U30%の task3_1 も LBP-U40%のプロセッサ 3 で稼働を続けることにより，時間制約は満たしているが仕事量が平準化されない（図 3(a-2)）．

一方，不均等型仕事量平準化方式では，マイグレーションを続けることで仕事量が平準化され，実行公平性が保たれていると言える（図 3(b)）．

タスクセット 2，タスクセット 3 も同様に，均等型仕事量平準化方式では，貼りつき問題により仕事量が平準化されないが，不均等型仕事量平準化方式では仕事量が平準化された．

以上より，不均等型仕事量平準化方式は，性能が不均等なマルチプロセッサ環境上でも，LB タスクの実行公平性を実現することができる可能性が高いといえる．

6. 関 連 研 究

マルチプロセッサ上でアプリケーションが資源を有効活用するための，さまざまな方法が提案されてきた．

例えば，文献 6)7) では，SMP 型 OS に対してリアルタイム性が求められるアプリケーションと，非リアルタイム性アプリケーションが稼働するプロセッサを区別することで，リアルタイム性とスループットの向上を実現する手法が研究されている．しかし，本研究では，プロセッサごとに HR タスクが割付けられるという使用方法を想定しており，HR タスクと LB タスクの稼動するプロセッサを区別しないことを前提とする．

7. お わ り に

本稿では，性能が不均等な組込み向けマルチプロセッサ用 RTOS 上で動作するマルチストリーミング処理に適したロードバランス機構として，不均等型仕事量平準化方式を提案し，実現した．その結果，不均等型仕事量平準化方式は均等型仕事量平準化方式と比較して，実行公平性を実現できる可能性が高いことを確認した．

なお，本研究では LB 判断値は，各 LB タスクの 1 フレームごとに加算する仕事量の最大値としたが，アプリケーションによって最適な値が存在すると考えられる．その最適値を算出する指針を検討することは今後の課題である．

謝辞 なお，本研究は，一部，科学研究費補助金（課題番号 23700035）の支援による．

参 考 文 献

1) 高田広章, 本田晋也,”機能分散マルチプロセッサ向けのリアルタイム OS,” 情報処理,47(1),pp.41-47,Jan,2006.
2) 本田晋也, 高田広章,”ITRON 仕様 OS の機能分散マルチプロセッサ拡張,” 電子情報通信学会論文誌, D, ,J91-D(4),pp.934-944,Apr.2008.
3) 相庭裕史, 本田晋也, 高田広章,”対称型マルチコアシステムのエンジン制御ソフトウェアへの適用,” 情報処理学会論文誌,Vol.51,No.12, pp.2238-2249,Dec.2010.
4) 石田利永子, 本田晋也, 高田広章, 福井昭也, 小川敏行, 田原康宏, ”組込み向けマルチプロセッサ対応 RTOS におけるタスク間の実行公平性を実現するロードバランス機構,” 組込みシステムシンポジウム 2010 （ESS2010）, pp. 97-105, Oct 2010.
5) 笠野英松, マルチメディアストリーミング技術,CQ 出版社, 東京,2011.
6) Philippe Marquet,Julien Soula,Eric Piel,and Jean-Luc Dekeyser, ”An asymmetric model for real-time and load-balancing on Linux SMP,” Research Report,2004-04,Laboratoire d’informatique fondamentale de Lille, Universit´e des sciences et technologies de Lille,France,Apr.2004.
7) Eric Piel,Philippe Marquet,Julien Soula,and Jean-Luc Dekeyser, ”Load-balancing for a real-time system based on asymmetric multi-processing,” Research Report 2004-06, Laboratoire d’informatique fondamentale de Lille, Universit´e des sciences et technologies de Lille,France,Apr.2004.