プロセッサアフィニティを考慮した Realtime Linux によるロボットの実時間制御

Real-Time Robot Control Using Realtime Linux under Consideration of Processor Affinity

○学　羽鳥 裕樹　（都市大）　学　加藤 龍一　（都市大）　学　白井 伸明　（都市大）
正　佐藤 大祐　（都市大）　正　金宮 好和　（都市大）

Yuki HATORI, Tokyo City University, hatori@rls.mse.tcu.ac.jp
Ryuichi KATO, Tokyo City University
Nobuaki SHIRAI, Tokyo City University
Daisuke SATO, Tokyo City University
Yoshikazu KANAMIYA (D. N. Nenchev), Tokyo City University

We address the real-time robot control using Linux. The RT-Preempt Patch [1][2] renders Linux with a fully preemptible kernel. We call the system Realtime Linux. The advantages of Realtime Linux is that we can use most of the Linux libraries and create a real-time program API based on POSIX. The aim of this paper is confirming multithread real-time robot control using a multicore processor, considering processor affinity, i.e. each task is allocated to kin processor in preference to others at allocation time. More specifically, we consider an eye-in-hand robot system comprising a PA–10 robot arm and a USB camera under real-time control with a multicore processor by efficiently assigning the processor affinity. Results are demonstrated with data from a face tracking experiment.

Key Words: Realtime Linux, Linux RT-Preempt, Real-time control, Processor affinity, Multicore processor

1　緒言

ロボットシステムを確実に制御するためには，ハードリアルタイムでの実時間制御が必要不可欠である．現在，国内で利用されているリアルタイム OS（RTOS）には，有償の VxWorks や QNX，無償の RTLinux や ART-Linux などがある．その中でも我々は，ロボットを制御する RTOS として Realtime Linux（RT-Preempt Patch を適応させた Linux）に注目している [1][2]．これまで我々は，Realtime Linux を用いたロボットの実時間制御を検証してきた．その検証結果より，シングルコアプロセッサによるロボットの実時間制御における有用性が確認されている [3][4]．しかし，近年ではマルチコアプロセッサが主流であり，これからのロボット制御にはマルチコアプロセッサを活用することが必要になる．そして，そのためにスレッドをプロセッサに割り当てるプロセッサアフィニティを考慮しなければならない．

そこで本稿では，プロセッサアフィニティを考慮した Realtime Linux による実時間制御について述べる．その中で，マルチコアプロセッサを用いたマルチスレッドによる実時間制御を確認し，スレッドをプロセッサ単位で分けることによる利点を示す．また具体例として，PA–10 ロボットアームと USB カメラを利用した eye-in-hand システムによる顔追従制御を実現し，その結果を示す．

2　Realtime Linux による実時間制御

システムがリアルタイムであるという場合は，要求された処理が終了するまでの時間に最低限の期限（デッドライン）を持つという意味である．また，要求の発生から応答を返すまでの時間をレイテンシという．このレイテンシと処理時間を合わせた時間がデッドライン以内であれば，リアルタイムシステムは正常であるといえる．

Realtime Linux とは，Linux のカーネルに高精度なリアルタイム性を持たせるために，RT-Preempt Patch を適応させた Linux のことである．Linux が利用できる多くのライブラリを用いることができ，また他の RTOS のような専用の API を用いることなく，POSIX 準拠の API のみでリアルタイムプログラムを作成することが可能であり，通常の Linux では不可能な 1 ms 以下の実時間制御を実現できる．

2.1　優先度

Linux において，すべてのプロセスは静的優先度（static priority）を持つ．通常のプロセスは常に 0 であり，リアルタイムプロセスには 1 以上 99 以下の値を与える．この静的優先度に従ってプロセスはプロセッサを割り振られ，Realtime Linux はプロセスの静的優先度に基づくプリエンプションを厳密に行い，リアルタイム性を確保する．

また，Realtime Linux では割り込みもスレッド化されており，IRQ（ハード割り込み要求），softirq（ソフト割り込み要求）をすべてカーネルスレッドに変換している [5]．そのため，図 1 (a) に示すような，高い静的優先度のスレッドで行われる処理中に生じた，IRQ による割り込みの逆転が起こる時間を最低限に抑えることができる．これは，図 1 (b) に示すように，割り込みが生じた際に割り込みサービススレッドを起こすのみで一度処理を戻す．その際，元のスレッドよりも高い静的優先度を持つ場合は，スケジュールさ

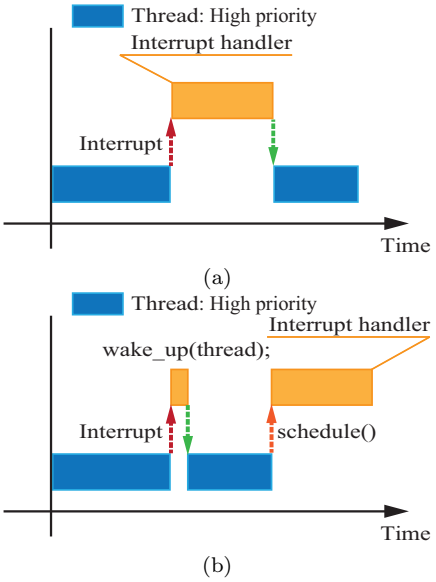


Fig. 1: Interruption processing: (a) The inversion of interruption, (b) Threaded interruption processing.

れて割り込みハンドラが実行される．逆に元のスレッドよりも低い静的優先度を持つ場合は，別のプロセッサにスケジュールされるか，高い静的優先度のスレッドの処理が終了するのを待つ．

よって，リアルタイムプログラムを作成する際は，作成するスレッドと割り込み要求の静的優先度を考慮することが必要である．

2.2 プロセッサアフィニティ

周期実行時のプロセススケジューリング例を図 2 に示す．この図は，周期実行する静的優先度の高いスレッドと静的優先度の低いスレッドの処理を表している．プロセッサが一つしかない場合は，静的優先度の高いスレッドが決められた周期毎に処理を行い，処理が終了すると静的優先度の低いスレッドが処理を再開する．そのため，スレッドが切り替わる毎にレイテンシが発生し，静的優先度の低いスレッドは処理を中断させられてしまう．それに対してマルチコアプロセッサでは，スレッドをプロセッサ毎に分けることが可能である．プロセッサアフィニティとはプロセスが同じプロセッサ上で実行する程度を表す．Linux のスケジューラは，同じプロセスを可能な限り同じプロセッサ上で実行させようとし，負荷のバランスが極端に悪くなった場合にのみ，プロセスを他のプロセッサに移動させる．この動作により，プロセッサ間の移動によるキャッシュへの悪影響を最低限に抑え，プロセッサへ均等な処理の配分が可能となる．

しかし，実時間制御時にプロセスがプロセッサ間を移動することは好ましくない．そのため，事前にスレッドとプロセッサの対応を固定させることにより，安定した処理が可能となる．また，図 2 (b) のように，それぞれのプロセッサにスレッドを割り当てることにより，静的優先度の低いスレッドは中断なく処理を実行でき，周期実行される処理とプリエンプションを考慮していない処理を同時に実行することも可能である．

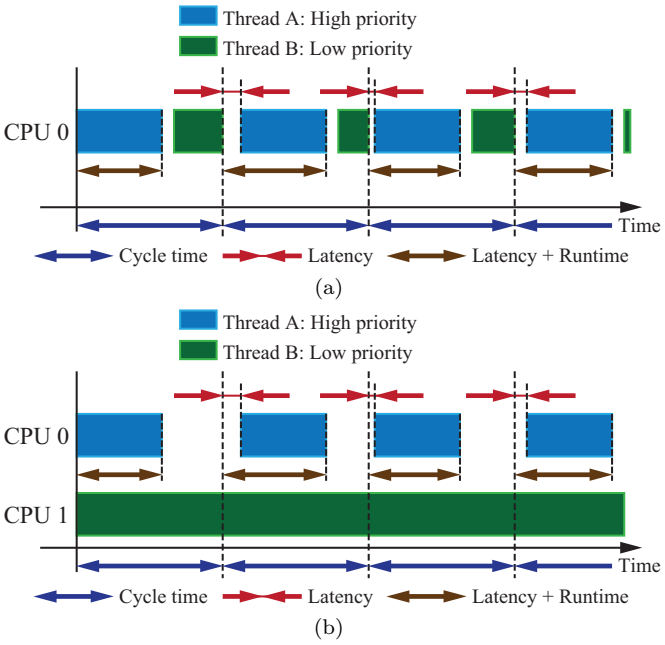


Fig. 2: Examples of process scheduling: (a) Single core processor, (b) Multicore processor.

3 プロセッサアフィニティ確認実験

実際のロボットを制御する前に，Realtime Linux においてマルチスレッドの処理とプロセッサアフィニティの関係を実験により確認した．1 ms で周期実行を行うスレッドと画像処理を行うスレッドの二つを用い，1 ms 以下の制御と静的優先度の低い画像処理についての結果を示す．

3.1 実験内容

制御用コンピュータとして ADVANTECH 社製の PCI-7020 を用いた．デュアルコアプロセッサ（Core 2 Duo E7400）を利用しているため，CPU0 と CPU1 の二つのプロセッサを持つ．また，画像取得のための USB カメラには，株式会社 Logicool 社製 Web カメラ QCAM-E3500 を用いた．

本実験では二つのスレッドを用いた．一つは 1 ms 周期で実行する周期実行スレッドで，0.5 ms のウェイト処理を行うスレッドである．もう一つのスレッドは，Linux で用いることのできる Open Computer Vision Library（OpenCV）2.0 を利用した，顔認識を行う画像処理スレッドである [6]．各スレッドの静的優先度は周期実行スレッドを 99，画像処理スレッドを 98 と設定した．

プロセッサアフィニティを利用した場合の状態を確認するための実験条件として以下の三つを設定した．

Case 1　プロセッサアフィニティの割り当てをプログラマが行わず，カーネルのデフォルト設定により制御

Case 2　プロセッサアフィニティの割り当てにより，CPU0 で二つのスレッドを実行するように設定して制御

Case 3　プロセッサアフィニティの割り当てにより，CPU0 で周期実行スレッド，CPU1 で画像処理スレッドを実行するように設定して制御

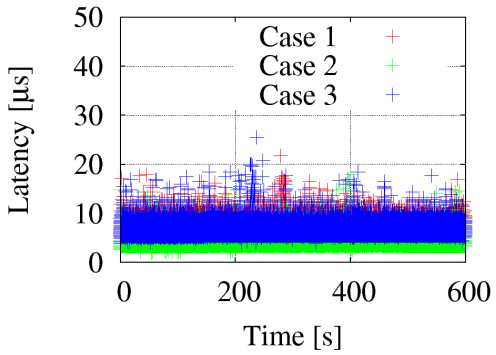
この条件の下で実験は 1 回の実験時間を 10 分として 10 回ずつ行い，周期実行スレッドのレイテンシおよびレイテンシと実行時間の合計値を測定した．

3.2 実験結果と考察

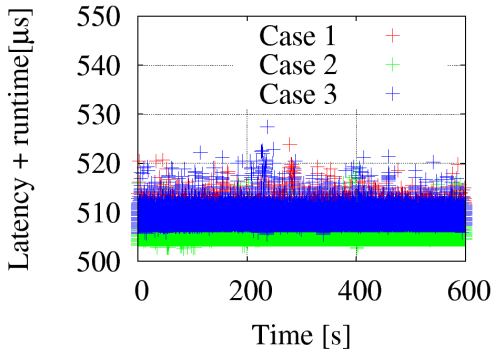
10 回行った実験結果の中で，それぞれレイテンシの最大値が最も大きかったデータ毎に，レイテンシおよびレイテンシと実行時間の合計値の結果を図 3 に示す．また，それぞれの最大値と平均値を表 1 に示す．図 3 と表 1 より，マルチコアプロセッサを用いたマルチスレッドによる制御において，1 ms 以下の制御が可能であること，スケジューラが静的優先度を厳密に判断していることが確認できた．

Case 1 と Case 2 を比較すると，レイテンシの最大値，平均値共に Case2 方が小さいことが分かる．また，Case 2 と Case 3 を比較すると，レイテンシの最大値，平均値共に Case2 方が小さいことが分かる．しかし，Case 1 と Case 2 では，取得した JPEG データの破損を意味するエラーが発生した．これは，画像処理スレッドで JPEG データの処理中に，同じプロセッサ上で静的優先度の高い周期実行スレッドに処理を奪われたことによって，エラーが発生したと考えられる．

これらの結果より，Case3 のように明示的にプロセッサを割り当てることで，プリエンプションを考慮していない処理と他の処理との干渉を抑えることが可能になる．



(a)



(b)

Fig. 3: Experimental results: (a) Latency, (b) Latency + runtime.

Table 1: Latency and latency + runtime.

Case	Latency	Latency + runtime
	Maximum (average) [μs]	
1	21.70 (6.02)	523.81 (508.34)
2	17.88 (3.85)	518.80 (505.02)
3	25.51 (10.50)	527.38 (508.17)

4 実機制御実験

前章の Case 3 のプロセッサアフィニティの設定を用いて，実際のロボットアームとカメラを用いた顔追従制御実験を行った．

4.1 実験内容

4.1.1 ハードウェア構成

実験に用いるロボットシステムを図 4 に示す．本実験では，三菱重工業株式会社製 7 自由度マニピュレータ PA–10 の手先に USB カメラを取り付けて実験を行った．本実験システムでは，PA–10 コントローラとの ARCNET に ISA バスを用いるため，ISA バスを搭載した PA–10 制御用コンピュータを用いた．

PA–10 制御用コンピュータでは，PA–10 コントローラと ARCNET を用いて通信を行う．また，動作生成用コンピュータでは，USB カメラと USB2.0 により通信を行い，PA–10 制御用コンピュータと 100BASE の Ethernet を用いたソケット通信を行う．

4.1.2 ソフトウェア構成

本実験システムのソフトウェア構成は，二つのルーチンに分かれている．制御内容をそれぞれのルーチン毎に説明する．

PA–10 制御ルーチン

PA–10 制御ルーチンは，PA–10 の動作を生成し，2.5 ms 周期で PA–10 コントローラへ指令値を送信し，エンコーダの値を取得する．このルーチンは，PA–10 制御用コンピュータの周期実行スレッドと動作生成用コンピュータの動作生成スレッドによる，二つのコンピュータ上のスレッドから構成される．

まず，動作生成スレッドにおいて PA–10 の動作生成を行い，PA–10 制御用コンピュータへソケット通信を用いて指令値を送信する．本実験では，後述する画像処理ルーチンから，カメラ画像の中央に顔が来るように手先位置の動作を生成する．そして，周期実行スレッドにおいて 2.5 ms 周期で PA–10 コントローラに ARCNET を用いて指令値を送信し，エンコーダの値を

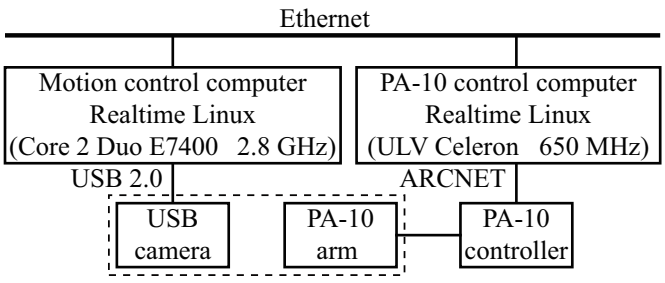


Fig. 4: Overview of the eye-in-hand robot system.

取得する．その後，動作生成スレッドとの間でソケット通信により，エンコーダの値を送信する．また，スレッドの静的優先度は両スレッド共に 99 と設定した．

画像処理ルーチン

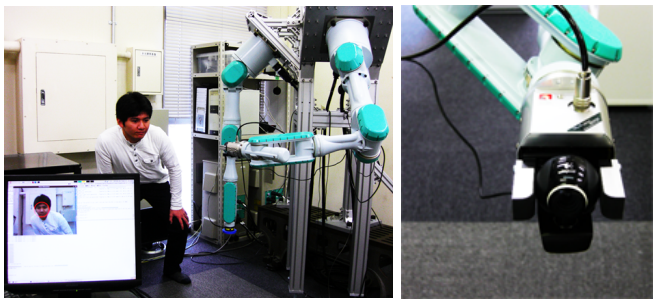
画像処理ルーチンは，カメラから取得される画像データより，目標対象物の位置を算出する．このルーチンは，動作生成用コンピュータの画像処理スレッドにより構成される．

この画像処理スレッドでは OpenCV 2.0 を用いて顔認識を行い，顔の位置を検出する．このスレッドで更新される最新の位置情報を用いて PA–10 制御ルーチンにおいて PA–10 の動作を生成，制御する．また，スレッドの静的優先度は 48 と設定した．このように設定したのは，ネットワーク関連の IRQ と softirq の静的優先度が，それぞれ 50 と 49 に設定されているためである．そのため周期実行スレッドで用いるソケット通信を考慮して，この割り込みの優先度よりも低い値として設定した．

以上の実験条件の下でロボットの顔追従制御を行い，周期実行が行われているか確認するために，周期実行スレッドのレイテンシおよびレイテンシと実行時間の合計値を測定した．

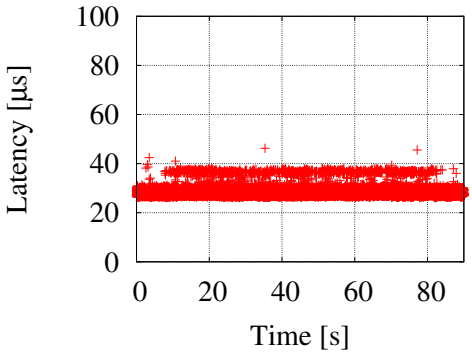
4.2 実験結果

図 5 に実験の様子を示す．実験により，ロボットアームの手先が被験者の顔を追従することを確認した．その際の周期実行スレッドのレイテンシおよびレイテンシと実行時間の合計値を図 6 に示す．また，それぞれの最大値と平均値を表 2 に示す．図 6 と表 2 より，周期内で処理が行われていることが分かる．これより，動作生成スレッドと周期実行スレッドにおいて周期的な通信が行われ，制御できていることが分かった．この実験結果から，マルチコアプロセッサにおけるマルチスレッドによるロボットの制御が可能であることが分かった．そして，プロセッサアフィニティを割り当てることにより，周期実行を行う処理と OpenCV2.0 を用いた処理をプロセッサ単位で分け，ロボットの実時間制御に利用できることが分かった．

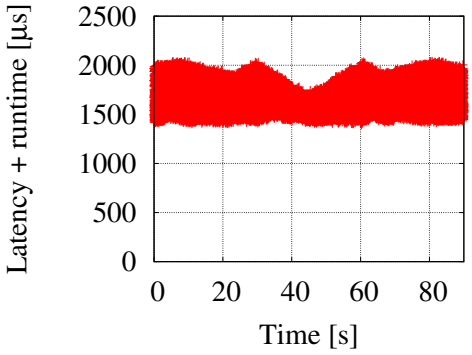


(a) (b)

Fig. 5: Eye-in-hand robot system: (a) The situation of a face tracking control experiment, (b) Enlarged view of eye-in-hand robot.



(a)



(b)

Fig. 6: Experimental results of face tracking control: (a) Latency, (b) Latency + runtime.

Table 2: Latency and latency + runtime.

Latency	Latency + runtime
Maximum (average) [μs]	
46.25 (28.33)	2038.48 (1647.68)

5 結言

Realtime Linux におけるマルチコアプロセッサを用いた，マルチスレッドによる実時間制御の影響を検証した．その結果，プロセッサアフィニティを割り当てることにより，周期実行を行う処理とプリエンプションを考慮していない処理をプロセッサ単位で分けた，ロボットの実時間制御を実現できた．

文 献

[1] (2010, March. 10) Real-Time Linux Wiki [Online]. Available: `http://rt.wiki.kernel.org/index.php/Main\_Page`

[2] (2010, March. 10) Open Source Automation Development Lab [Online]. Available: `http://www.osadl.org/Realtime-Linux.projects-realtime-linux.0.html`

[3] S. Kume, Y. Kanamiya, and D. Sato , “Towards an open–source integrated development and real–time control platform for robots,” *in Proc. of the 2009 IEEE Int. Conf. on Robotics and Biomimetics,* Bangkok, Thailand, MoA6.6, Feb. 22-26, 2009.

[4] 白井伸明，羽鳥裕樹，佐藤大祐，金宮好和，“Linux with RT-Preemptible Patch を用いたロボットシステムの実時間制御”，第 27 回日本ロボット学会学術講演会予稿集 CD-ROM ，3F2-07, 2009.

[5] Karim Yaghmour, Jon Masters, Gilad Ben-Yossef, Philippe Gerum (水原文 訳), “組み込み LINUX システム構築 第 2 版”，株式会社オライリー・ジャパン, 2009, pp. 393–426.

[6] (2010, March. 10) OpenCVWiki [Online]. Available: `http://opencv.willowgarage.com/wiki/`