Linux におけるリアルタイムプリエンプションパッチに基づいたロボットのリアルタイム制御

Real-Time Robot Control Based on the Linux Real-Time Preemption Patch

○ 久米 修平 （武蔵工大）　正 金宮 好和 （武蔵工大）　正 佐藤 大祐 （武蔵工大）

○ Shuhei Kume, Musashi Institute of Technology, kume@rls.mes.musashi-tech.ac.jp
Yoshikazu Kanamiya (D. N. Nenchev), Musashi Institute of Technology
Daishuke Sato, Musashi Institute of Technology

The Linux real-time preempt kernel patch, developed by Ingo Molnar, has recently drawn attention as an alternative to other dedicated real-time OSs. The patch implements real-time behavior by allowing nearly all of the kernel to be preempted, and by usage of high resolution timers. We describe our experimental environment for real-time control of a PA-10 robot arm and present experimental data for real-time capability of the real-time preempt kernel patch based controller.

Key Words: Real-Time Control, Linux RT-Preempt.

1　序論

現在ロボットを用いた研究は，ヒューマノイドロボットを人間環境へ適応させる研究を始め，様々なロボットを様々な形で利用する研究がされている．ロボットの研究を始める際，対象となるロボットを制御するため OS を選定する必要性がある．市販のロボットの場合，制御用のアプリケーションと OS が対になっている場合が多く，OS を選定する必要はない．しかし自前でロボットを制御したい，または市販のロボットをカスタマイズしたい場合，付属のアプリケーションでは事足りない場合がある．さらにロボットの制御には，周期的な制御を行う必要があるため，扱うロボットによっては厳密なリアルタイム性が要求される．そこでリアルタイム性を実現するリアルタイム OS（RTOS）が必要となる．

現在，無償有償を含めて，VxWorks[1]，QNX[2]，μITRON[3]，国産 RTOS の ART-Linux[4]，RTLinux，Xenomai[5]，RTAI[6] などがある．また無償である Linux に対してリアルタイム性を持たせる取組みとしては，通常の kernel のみで 1 ms まで実現可能であることが確認されている [7][8]．これらに加えて，Linux の kernel に更に高精度なリアルタイム性を持たせるため，プロセスの優先度付けを厳密に行うことでリアルタイム性を確保する RT preemption patch（以下 RT preempt とする）が開発された [9]．これを用いた取り組みとして，AV 関連 [10]，ペイントロボット [11]，ウェアラブルコンピュータ [12] などが挙げられる．

本稿ではこの RT preempt を Linux マシンに当て，研究対象である三菱重工業社製 7 自由度マニピュレータ PA-10-ARM（以下 PA-10 とする）へ用いた際のリアルタイム性の検証について述べる．

2　RT preemption patch

2.1　リアルタイムの概要

リアルタイム処理は，一般的に処理を切り替えるために費やす最大時間，つまりワーストケースのレイテンシ（待ち時間）がどの程度保証できるかで大別される．100 % 保証できるものを「ハードリアルタイム」，100 % まではいかないが，高い確率で規定した時間内に収まるものを「ソフトリアルタイム」と呼ぶ．

あるプロセスが実行中に，OS への自発的な CPU 返上を待たずに，該当プロセスから CPU を強制的に切り替えることをプリエンプション（優先度付け）といい，多くの OS には標準に搭載されている機能である．リアルタイム処理で最も重要なことは，現在実行している処理よりも優先すべき処理が見つかった場合，速やかに切り替えることである．つまり，どんな場合でも遅延なくプリエンプションを行えることである．

2.2　推移

Kernel のプリエンプションについて，2.4 までは機能が搭載されていなかった．この 2.4 にプリエンプションを搭載されたものが，preemtible kernel 2.4 である．これ以降，バージョンがあがるにつれ高性能なプリエンプションへと改善がなされている [10].

Kernel のプリエンプションに関して，kernel 2.4 では kernel の内部コードを実行中は，割込みなどを契機にプリエンプションを行うことができなかった．このとき，kernel の内部コードからユーザープログラムのコードに復帰する際に再スケジューリングをし、他のプロセス／スレッドに処理を切り替えていた．

kernel 2.6 は標準の状態で preemptible kernel であり，kernel の内部コードにプリエンプションを行うポイントを設け，ユーザープログラムのコードに戻る地点よりも早く

に優先度（priority）の高い処理に切り替えることを可能とした．これにより kernel 2.4 よりレイテンシを改善することが可能となった．

しかし，この kernel では長期間スピンロック[*1]を解放しない区間が存在する．この区間を細粒化したのが，2004 年 7 月に Ingo Molnar（Red Hat）により提案された Voluntary kernel preemption である [13]．クリティカルセクション中にすでに含まれている `might_sleep` というデバッグ用のマクロ関数をスケジューリング・ポイントとすることで，この関数が処理されて条件が満たされれば，スピンロックは破られ優先度の高い処理へ切り替えられる．これによりシステム全体のレイテンシを低減した．これは kernel 2.4 に対して提案されたロー・レイテンシ・パッチ（Low Latency Patch）に相当するものである．

さらに Molnar は，Voluntary kernel preemption の抜本的な見直しを継続し，2004 年 10 月 RT preemption patch をリリースした [9]．特徴および Voluntary kernel preemption から追加された機能は以下の通りである．

- 多様なアーキテクチャにも対応(x86, x86-64, ARM[*2], MIPS[*3])
- RT preempt の API はなく，標準の POSIX API でリアルタイムのプログラムが書ける
- 割込みハンドラのコード大部分をカーネルスレッドにより実行する
- セマフォ[*4]を用いてスピンロックする
- セマフォに優先度継承機能を適用
- スピンロックによるクリティカルセクションの見直し

2.3 検証

RT preempt を用いてリアルタイム性の検証を行った．ハードおよび OS，kernel は以下の通りである．なお X windows を始め，出来る限りのデーモンを切り，メモリ使用率の軽減を行った．

- アドバンテック社製 ISA バス SBC: PCA-6773（Intel Cerelon ULV 650 MHz，SODIMM 256 MB）
- CentOS 5（デフォルトは kernel 2.6.18 だが，kernel 2.6.23 に変え，patch-2.6.23-rt1 をサイトからダウンロードし再構築 [14]）

検証に用いたプログラムは，パラレルポートの 1 ポートに 100 μs 周期で出力するサンプルプログラム（Squarewave-example）である [9]．負荷を掛けた状態での非リアルタイム，リアルタイムで実行し，CPU タイマを用いて計測を行った．リアルタイムの場合，プロセスの priority を 90 とした．なお priority は 0 から 99 までの間で設定可能である．

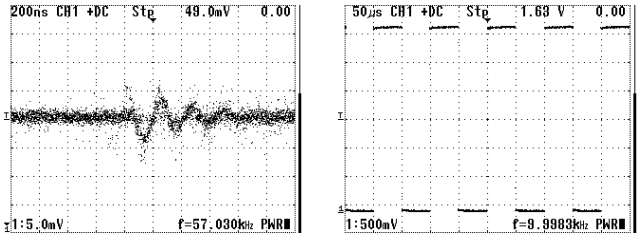
100 μs 周期には以下の関数を用い sleep を掛けた．

```
clock_nanosleep(0, TIMER_ABSTIME, &t,
NULL);
```

負荷を掛けたコマンドは以下のコマンドである．

```
# cat /dev/zero > delete_this_file
```

Fig.1 に出力された矩形波を示す．



(a) non real-time (b) real-time

Fig.1: Square wave with load.

(a) では非リアルタイムであるため，パラレルポートに対して 100 μs 周期で出力しているが，負荷を掛けているため波形は乱れていることが確認できる．一方，(b) ではリアルタイム下で出力を行っているため，100 μs の周期で安定かつ正確な波形を出力しているのが分かる．よって RT preempt の有用性は確認できた．

3 RT preempt を用いた PA-10 のリアルタイム制御

3.1 概要

研究では実験機として PA-10 を用いている．先の検証に用いた PCA-6773 に ARCNET のボードを取り付け制御 PC とした．PA-10-CNT の通信方式は ARCNET [*5]を用いており，制御周期は 2 ms である．この実験機を用いて，RT preempt のリアルタイム性の検証を二通り行った．Fig. 2 に実験システムを示す．

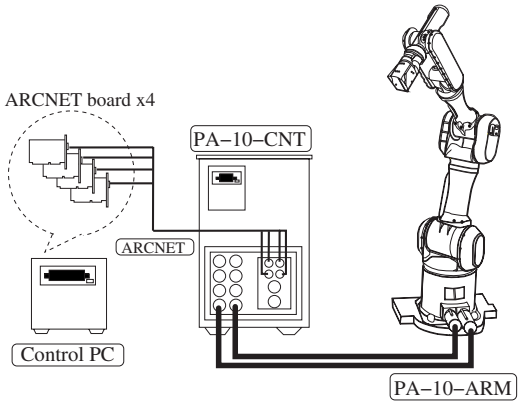


Fig.2: Experimental system.

[*1] スレッドがロックを獲得できるまで単純にループ（スピン）して定期的にロックをチェックしながら待つ方式．

[*2] Acorn RISC Machine：組み込み機器や低電力アプリケーション向けに広く使われている 32 ビット RISC CPU のアーキテクチャ．

[*3] Microprocessor without Interlocked Pipeline Stages：NINTENDO64，PS，PS2 などで採用されているアーキテクチャ．

[*4] 並列処理の実行環境において，排他区間を確保し，資源に同時アクセスできる上限を規定したい時に用いる．

[*5] 1997 年米国 Datapoint 社によって提唱された改良型トークンパッシング・プロトコルの LAN．改良型トークン・パッシング方式を採用することによりリアルタイム性を保証．

3.2 実機制御による検証

RT preempt をインストールした制御 PC より制御周期 2 ms 毎に速度指令値を送信し，2 ms からのジッター（ずれ）を計測した．このとき，PA-10 第 6 軸のみをスプラインで軌道生成し，0 deg から 45 deg まで動作させ，目標関節角度到達以降は停止させ，ジッターの様子を観察した．なお総動作時間は 300 s である．その結果を Fig. 3 に，Table. 1 に最大値，平均，標準偏差を示す．

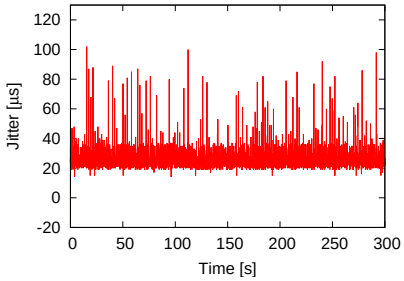


Fig.3: Jitter under control environment with RT preempt.

Table1: Various values of the jitter in Fig. 3.

Max	Average	Standard deviation
102 μs	25.1 μs	7.57 μs

patch を適用していない標準の kernel では，1 ms の制御周期が限界である [8]．しかし Fig. 3 に示すように，RT preempt をインストールしたリアルタイム下においては，2 ms 周期毎に小さいジッターで安定した出力をしているのが確認できる．

3.3 MaTX を用いた検証

本研究室において，制御シミュレーションには MaTX を用いている．MaTX とは，科学や工学に必要な数値および数式計算をサポートする記述性に優れた C 言語ベースで開発されたプログラミング言語であり，1989 年に古賀雅伸が開発し，制御系の解析・設計・シミュレーションの実行に利用されているフリーのソフトウェアである．主な特徴として，以下の点が挙げられる [15]．

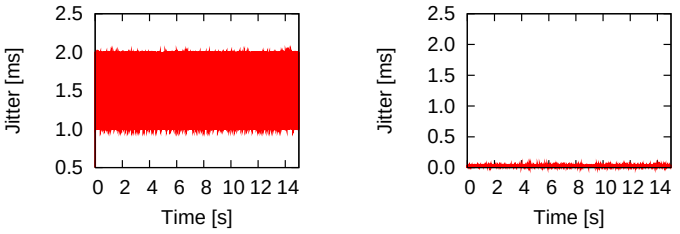
- 行列，多項式，有理多項式，多項式行列，有理多項式行列などの型が標準で用意されており，アルゴリズムを簡潔に表現可能
- C 言語のプログラムをリンクできるので，C で書かれたソフトウェア資産を活用可能
- リアルタイム制御用プログラムを直接記述可能

この MaTX を PA-10 制御 PC にインストールし，MaTX プログラムからリアルタイムプログラムを呼び出し，ジッターを計測した．MaTX からリアルタイムプログラムを呼び出すことで，シミュレーションなどのプログラムを実機に適用する際，プログラムの大幅な変更を回避できる．C で書かれたプログラムと MaTX プログラムを共にコンパイルする際には，以下のコマンドが必要である．

```
$ matc -v main.mm pa10.c -lrt
```

- matc：MaTX のコンパイラで MaTX で記述されたファイルを C へ変換し，実行ファイルを作成
- main.mm：MaTX で書かれた main ファイル
- pa10.c：リアルタイム文を記述した C 言語ファイル
- -v：コンパイルの各ステージで実行されるコマンドを表示するオプション
- -lrt：リアルタイムのライブラリをリンクするオプション

先に述べた環境下でプログラムを実行したときの非リアルタイム，リアルタイムの場合を Fig. 4 に結果を示す．なお計測時間は 15 s である．この結果は，PA-10 の制御周期と合わせて 2 ms 毎の時間を保存し，2 ms からのジッターを求めた．



(a) non real-time　　(b) real-time

Fig.4: Running MaTX program under RT preempt.

2 ms 毎に時間を計測しているが，(a) では非リアルタイム下であるため，少なくとも 1 ms，大きい場合は 2 ms のジッターが確認できる．一方，リアルタイム下である (b) ではほとんどジッターがないことが確認できる．この拡大図を Fig. 5 に，また (a)，(b) のジッターの最大値，平均，標準偏差を Table. 2 に示す．

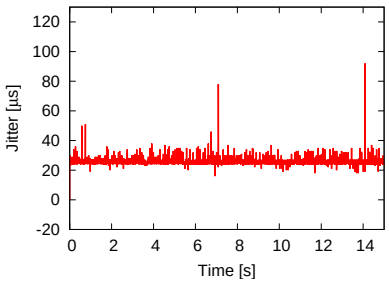


Fig.5: Zoom in Fig. 4 (b) data.

Table2: Various values of the jitter in Fig. 4.

	Max	Average	Standard deviation
non real-time	2.06 ms	1.45 ms	0.49 ms
real-time	92.0 μs	25.5 μs	1.82 μs

Fig. 5 より，ジッターが大きくても 92.0 μs となり，(a) の non real-time の場合と比べ，リアルタイムでは高精度な結果が確認できた．また平均でも 25.5 μs であり，PA-10 を 2 ms の周期で制御する場合でも問題はないと言える．

4 結論

RT preempt を通常の Linux マシンにインストールすることで，高精度なリアルタイム性を確認することができた．

また実機制御および制御シミュレーションを行う MaTX と併用しリアルタイム性を確認することができた.

文　献

[1] (2008, March 21) Wind River [Online]. Available: http://www.windriver.com/

[2] (2008, March 21) QNX Software Systems [Online]. Available: http://www.qnx.co.jp/

[3] (2008, March 21) ITRON Project Archive [Online]. Available: http://www.ertl.jp/ITRON/home-j.html

[4] (2008, March 21) ムービングアイ [Online]. Available: http://www.movingeye.co.jp/

[5] (2008, March 21) Xenomai: Real-Time Framewaork for Linux [Online]. Available: http://www.xenomai.org/index.php/Main_Page

[6] (2008, March 21) RTAI - the RealTime Application Interface for Linux from DIAPM [Online]. Available: https://www.rtai.org/

[7] 熊谷正朗, 江村超：“汎用 Linux によるロボット制御システムの開発，” 日本ロボット学会誌，Vol.20 No.2，pp.157-163 (2002)

[8] 熊谷正朗, “汎用 Linux によるロボット制御 -KNOPPIX + kernel 2.6 による開発環境の構築-,” 日本機械学会ロボティクス・メカトロニクス講演会'06, 1P1-C40, 2006.

[9] (2008, March 21) Real-Time Linux Wiki [Online]. Available: http://rt.wiki.kernel.org/index.php/Main_Page

[10] Daniel Walker:“The Evolution of Real-Time Linux,” ftp://ftp.realtimelinuxfoundation.org/pub/events/rtlws-2005/SvenThorstenDietrich.pdf

[11] Morten Mossige, Pradyumna Sampath, Rachana G Rao: “Evaluation of Linux rt-preempt for embedded industrial devices for Automation and Power Technologies - A Case Stude,” http://www.linuxdevices.com/files/article081/Sampath.pdf

[12] Dongwook Kang, Woojoong Lee, Chanik Park: “Kernel Thread Scheduling in Real-Time Linux for Wearable Computers,” ETRI Journal, Vol. 29, No. 3, June 2007.

[13] (2008, March 21) Linux による RTOS の実現 (3) -スタンドアロン式とリアルタイム性能- [Online] Available: http://monoist.atmarkit.co.jp/fembedded/rtos03/rtos03c.html

[14] (2008, March 21) Index of /pub/linux/kernel/projects/rt [Online]. http://www.kernel.org/pub/linux/kernel/projects/rt/

[15] (2008, March 21) MaTX [Online]. http://www.matx.org/