

組込み機器における Linux カーネル 2.6 のリアルタイム性評価

出原 章雄[†] 攝津 敦[†] 伊藤 孝之[†] 落合 真一[†]三菱電機 (株) 情報技術総合研究所[†]

1. はじめに

デジタル家電や携帯電話などの組込み機器では、高機能化に伴い、メモリ保護やネットワーク機能などを備えた UNIX 系 OS、特に Linux の採用が増加しており、最近ではリアルタイム OS(以下 RTOS)が採用されていた、マイクロ秒オーダーの応答性を必要とする領域に対しても適用が広がってきている。従来、このような領域に対して Linux を用いる場合には、RTOS と Linux を動作させるハイブリッド方式等を用い、時間制約のある処理を、リアルタイム性をもつ別の機構で処理していた[1][2][3]。

しかし Linux カーネル 2.6 では、カーネルのプリエンプション機能がサポートされ、Linux 単体でのリアルタイム性が強化されている。また、オープンソースコミュニティでは、更なるリアルタイム性の強化を目指したプロジェクトがあり、その中で、オリジナルのカーネルに対するパッチが開発されている。そこで今回、これら Linux カーネル 2.6 のリアルタイム性について評価を実施した。本稿では、その評価結果について述べる。

2. プロセス応答時間測定

2.1. 測定環境

x86 ベースのボード上に各 Linux カーネルを構築し、この上で評価を実施した。評価対象の H/W およびカーネルを表 1、表 2 に示す。なお、各カーネルとも 1tick=10 ミリ秒である。

表 1 評価対象の H/W

CPU	x86 ベース 600MHz
メモリ	256MB
ファイルシステム	コンパクトフラッシュ ext2: 256MB (PIO)

表 2 評価対象のカーネル

カーネル	ベース	主なオプション、使用したパッチ
PRE_OFF	2.6.8	プリエンプション無し
PRE_ON	2.6.8	プリエンプション有り
PRE_VOL	2.6.8	ボランタリプリエンプトパッチ適用
PRE_RT	2.6.13	リアルタイムプリエンプトパッチ適用

ボランタリプリエンプトパッチおよびリアルタイムプリエンプトパッチ[4]は、Ingo Molnar 氏がメンテナとなって開発が進められている、Linux カーネルに対するリアルタイム性強化パッチであり、以下の機能を持つ。

● ボランタリプリエンプトパッチ (PRE_VOL パッチ)

■ボランタリプリエンプト：実行中のプロセスが、クリティカルセッション実行中にプリエンプ可能の場合、スケジューリングを行い、他のプロセスにスイッチすることにより、応答時間の短縮を図る。

■Threaded IRQ：H/W 割込み(以下 hardirq)ハンドラをスレッド単位で実施する。割込みコンテキストでは hardirq ハンドラ起床のみを行うため、割込み禁止で動作する、割込みコンテキストの実行時間が短縮される。

Evaluation of Linux kernel 2.6 as a "realtime" operating system for consumer device

[†]Akio Idehara, Atsushi Settsu, Takayuki Ito, Shinichi Ochiai
Information Technology R&D Center, Mitsubishi Electric Corporation

● リアルタイムプリエンプトパッチ(PRE_RT パッチ)

PRE_VOL パッチの機能に加え、以下の機能を持つ。

■優先度継承 mutex：スピンロックを、優先度継承 mutex で置き換える。これにより、スケジューラでのアトミック操作のような一部の例外を除いて、スピンロックで保護されている領域に対しても、プリエンプが可能となる。

■Threaded IRQ のチューニング：ソフトウェア割込み(以下 softirq)スレッド(以下 softirqd)及び hardirq スレッド(以下 hardirqd)をリアルタイム優先度に変更する。

2.2. 測定方法

負荷をかけた状態で高優先度プロセスの周期起動遅延を測定した。測定ポイントは以下の通り。

- ① タイマ割込みハンドラの先頭(図 1 ①)
- ② タイマ割込みハンドラから起動される、SIGALRM ハンドラの先頭(図 1 ②)

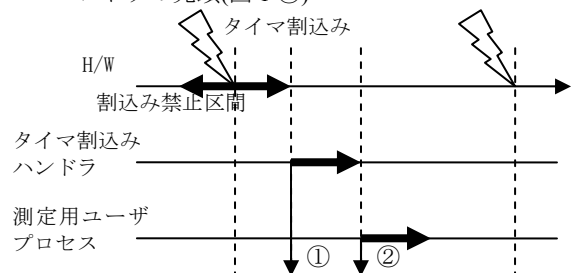


図 1 測定方法

これにより、以下の時間が求められる。

■割込み禁止時間：タイマ割込みハンドラ起動時刻(図 1 ①)について、1tick 毎のジッタ(起動時刻の差分)を算出(①-タイマ割込み通知時刻)

■プロセス応答時間：タイマ割込みハンドラ起動時刻(図 1 ①)とユーザプロセス起動時刻(図 1 ②)の差分を算出(②-①)

また、負荷ツールは、自製のツール類を用いている。

3. 測定結果

測定結果の一部として、共有メモリ生成/削除負荷(図 2、図 4)と、ブロックデバイス読み出し負荷(図 3、図 5)を示す。測定結果から以下のことが判明した。

3.1. 割込み禁止時間(図 2、図 3)

共有メモリ負荷では、各カーネルとも大きな違いは無い。また、ブロックデバイス負荷では、PRE_RT は、PRE_OFF と比較して、10 分の 1 の応答時間となり、大幅な改善が見られる。また、今回は示していないが、PRE_VOL は、PRE_OFF と比較して、半分程度の応答時間となり、改善が見られる。

3.2. プロセス応答時間(図 4、図 5)

応答まで十数ミリ秒かかっていたカーネル 2.4 と比較すると、各負荷とも、PRE_OFF の場合でも数ミリ秒程度で応答しており、改善が見られる。また、ブロックデバイス負荷について、PRE_RT では PRE_OFF と比較して、平均的な応答性能が向上している。しかし、最悪値は、1 ミリ秒以上となっており、PRE_RT を用いても、最悪値

の改善は見られない。

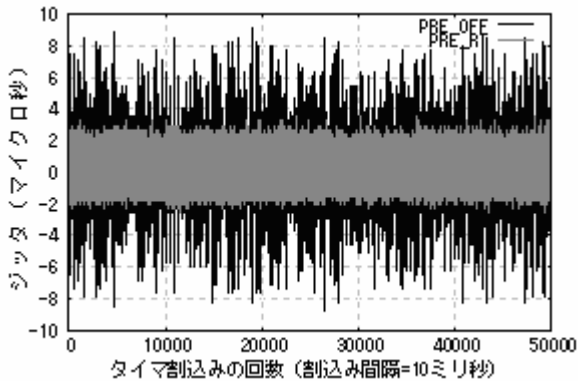


図2 共有メモリ負荷に対する割込み禁止時間

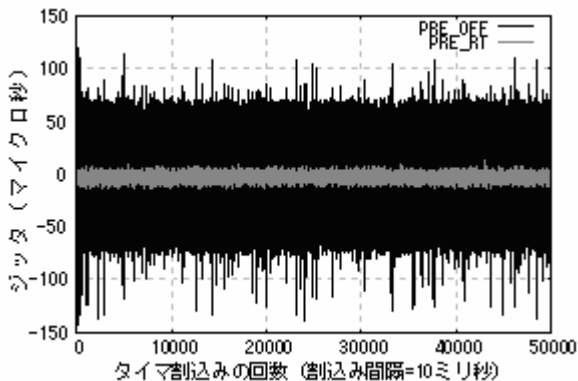


図3 ブロックデバイス負荷に対する割込み禁止時間

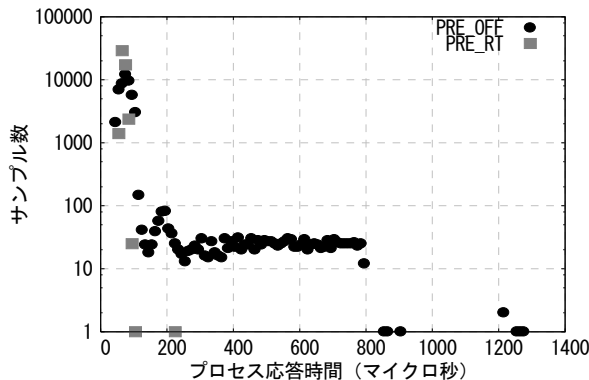


図4 共有メモリ負荷に対するプロセス応答時間

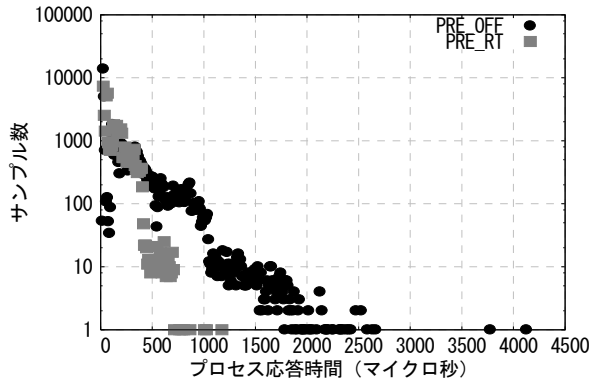


図5 ブロックデバイス負荷に対するプロセス応答時間

4. 考察

4.1. 割込み禁止時間(図2、図3)

共有メモリ負荷について、PRE_OFF の場合でも、割込

みコンテキストがクリティカルセッションで動作する時間が短いため、PRE_RT との差は見られない。

ブロックデバイス負荷について、PRE_OFF の場合、割込みコンテキストがクリティカルセッションで動作する時間が長い。これに対して、今回は示していないが、PRE_VOL の場合、ジッタは 60 マイクロ秒程度となり、Threaded IRQ の効果が現れている。また、PRE_RT の場合、優先度継承 mutex により、スピンロックで保護されている領域を実行中でも、H/W 割込みが許可されていれば、割込みコンテキストにスイッチ可能なため、割込み禁止時間が 10 マイクロ秒まで減少したと考えられる。

4.2. プロセス応答時間(図4、図5)

共有メモリ負荷について、PRE_OFF では 1 ミリ秒程度の時間が掛かっている。これは、PRE_OFF ではカーネルプリエンプシジョンが発生しないため、タイマ割込み発生後にカーネル内を走行する時間が最長で 1 ミリ秒程度であると考えられる。これに対して、今回は示していないが、PRE_ON では、一定のサンプル数となる区間が 500 マイクロ秒程度、PRE_RT では 200 マイクロ秒程度の応答時間となっており、カーネルプリエンプシジョンの効果が現れたものと考えられる。

4.3. プロセス応答時間の最悪値について

特に応答性の悪かった、ブロックデバイス負荷について PRE_RT を調査したところ、以下が判明した。

PRE_RT ではタイマ割込み以外の hardirq ハンドラをスレッド実行している。この hardirq スレッド(hardirqd)の優先度は、hardirq イベントをプロセスへ通知する softirq スレッド(softirqd)の優先度よりも高い。そのため、hardirq の発生頻度が極端に高い場合、softirqd にスイッチせず、hardirq が動作し続けてしまう。特に今回の測定では、優先度の低い負荷ツールの hardirq 処理に引きずられ、優先度の高い測定用プロセスの起動に遅延が発生していた。

また、softirqd 内部でも、ユーザプロセスへイベントを通知する前に、ルーティング用のハッシュテーブルを書き換える処理など、タイマに連動した処理が動作する。このような処理に長い時間を費やした場合、ユーザプロセスへのイベント通知が、タイマ処理に引きずられ、遅延してしまう。

このように、hardirqd スレッドの優先度、および、タイマ処理の長時間動作など、処理の優先度付けに改善の余地があることが判明した。

5. おわりに

本稿では、Linux カーネル 2.6 のリアルタイム性評価結果について述べた。評価の結果、Linux2.6 では、2.4 に比べリアルタイム性が向上し、さらにリアルタイムプリエンプトパッチを適用することで、平均的な応答時間が 1 ミリ秒以下に収まることが判明した。ただし、最悪値は 1 ミリ秒以下にはならなかった。これは、H/W イベントに対する優先度付けなど、今まで以上にリアルタイム処理を意識した改善が必要なためである。今後はこれら改善点について検討を行う予定である。

参考文献

- [1] RTLinux: <http://www.rtlinuxfree.com/>
- [2] RTAI: <http://www.rtai.org/>
- [3] Linux on RTOS:
http://www.emblinx.org/PDF/emblinx_hawg0301.pdf
- [4] リアルタイムプリエンプトパッチ:
<http://people.redhat.com/mingo/realtime-preempt/>