マルチコアCPUでのLinux(†)リアルタイム性向上手法

コアごとのタイマ処理制御で汎用処理の影響を回避

Linux(†)を組込み機器に適用する際，リアルタイム（以下，RTと略記）性が重要です。マルチコアCPUを搭載する組込み機器では，汎用処理とRT処理を区別してそれぞれ異なるコアで動作させます。これによってRT用コアが汎用処理の影響を受けないようにして，RT性を向上させることができます。しかし，OS（基本ソフトウェア）の処理の中には動作するコアを指定できないものがあり，これがRT性を更に向上させるうえで問題になっていることがわかりました。

そこで，コアを指定できない処理であってもRT用コアの実行負荷をなくす手法を考案し，マルチコアCPUにおけるLinux(†)のRT性を向上させました。

RT処理及び汎用処理混在時の課題

RT性とは，定められたイベントに対してアプリケーションが迅速に応答し，締切りまでに処理を完了できる性能のことで，組込み機器では特に強く要求されます。RT性を評価するためには，OSの割込み遅延時間を測定します。割込み遅延時間とは，ハードウェア（以下，HWと略記）が割込みを発生させた時点（**図1**のt_0）から，その割込みを必要とするアプリケーションが動作を開始する時点（t_1）までの時間です。この時間が平均的に，かつ最長時でも短ければ，そのOSはRT性が優れていると言えます。長い遅延時間は，コアが分割できない大事な処理を行うときに割込み禁止状態になることが原因で発生することがあります。

これに対し，締切りを持たない処理のことを汎用処理と呼びます。RT処理に比べて処理時間の予測が難しいことが特徴です。Linux(†)はもともと汎用処理向けであったために，RT性はあまり良くありません。しかし割込み禁止状態を短くするRTパッチを適用してRT性を向上させ，Linux(†)をRT処理向けに特化させることができます。

一方で，Linux(†)を利用する場合，CPUリソースを有効活用するために，汎用処理とRT処理を混在して動作させたいという要求が多くあります。このような要求に応えるため，RTパッチを適用したLinux(†)上で，RT処理を優先して動作させ，汎用処理の動作を後回しにする方法が一般に用いられています。しかし，たとえ後回しにしても，汎用処理が動作した際にファイル処理やネットワーク処理を行うことがあります。このような処理では，扱うデータ量に依存して割込み禁止状態の長さが変わります。このため，RT処理と汎用処理を混在させた場合，RTパッチを適用したLinux(†)でもRT性が悪くなるという課題があります。

CPU Shieldによる RT性の向上手法

この課題を解決するために，CPU Shieldと呼ばれる手法があります。マルチコアCPUを搭載するシステムでは，処理が行われるコアを指定するアフィニティ機能がLinux(†)に備わっています。アフィニティ機能を使ってRT処理を専用のRT用コアに割り当てる手法がCPU Shieldです。これにより，汎用処理による割込み禁止状態が汎用コアで長期間続いても，RT用コアではRT性への悪影響を受けません。

アフィニティ機能を適用できない処理

CPU Shieldを用いてRT用コアの割込み遅延時間を測定してヒストグラムにした結果を**図2**に示します。ほぼ全ての結果が100 μs以下ですが，一つだけ1,100 μs以上の割込み遅延時間が発生しています。今回，OSの処理にはアフィニティ機能を適用して動作するコアを指定できないものがあることがわかりました。更に解析の結果，そのような処理が長時間の割込み禁止状態を引き起こし，割込み遅延時間を大きくしている原因だということもわかりました。原因となったOSの処理はタイマ処理です。この処理は，コアごとに一つずつが固定されているため，アフィニティ機能が適用できません。また，以下に述べる複雑な処理が原因となってRT性を低下させます。

タイマ処理は，アプリケーションからの要求に応じて，登録された処理をある特定の時刻に実行します。どのような時刻の要求にも対応できるように，時間の粒度が異なる複数のタイマリストを用いてタイマ要求を登録しています（**図3**）。最初に実行するタイマ要求はリスト1に登録し，時刻が進んでいくに従って，上位のタイマリストからタイマ要求の付替えを定期的に行います。この付替え処理を行っている間は，タイマリストの一貫性を守るために割込み禁止状態にしています。これがRT性を低下させる原因です。

RT用コアのタイマ処理の負荷制御

そこで次のような手法を考案し，RT用コア上のタイマ処理を制御して付替え処理を行わなくてもよい機能を実現しました（**図4**）。

(1) RT用コアのタイマリストに既に登録された要求を，汎用コアのタイマリストへ移行する

(2) RT用コアへの要求の登録を禁止し，汎用コアに要求が登録されるようにする

(1)によってRT用コアのタイマリストを空にして，(2)によってその状態を保持します。

この機能を実装して割込み遅延時間を計測した結果を**図5**に示します。もっとも長い割込み遅延時間が60 μs以下となり，図2と比較してRT性がはるかに良いことがわかります。この手法によって，RT用コアでは長時間の割込み禁止状態が発生することがなくなり，RT性の向上を確認しました。

今後の展望

今回開発した機能をLinux(†)コミュニティで公開して広く意見を収集し，更なる改善点を検討します。

その後，この手法を生かしてRT性への要求が厳しくLinux(†)が適用できていない組込み機器への適用を拡大していきます。

† Linuxは，Linus Torvalds氏の日本及びその他の国における登録商標又は商標。

山田　真大
ソフトウェア技術センター
先端ソフトウェア開発担当主務

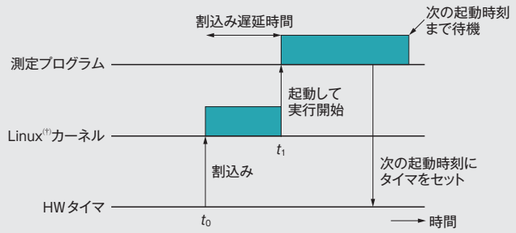

図1．割込み遅延時間の測定 — HWタイマを使って割込みを発生させ，割込み遅延時間を測定してLinux(†)のRT性を評価します。

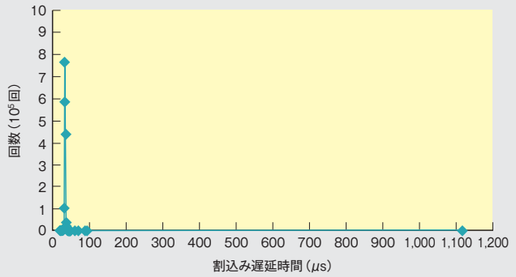

図2．タイマ処理によってRT性が低下した場合の測定結果 — 1,100 μs以上の割込み遅延時間となるケースが発生しており，RT性が悪いことを示しています。

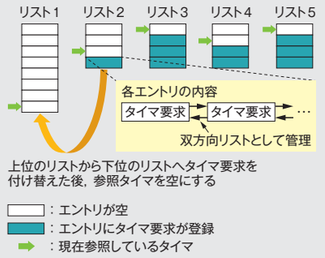

図3．タイマ処理が管理している複数のタイマリスト — 粒度の異なるリストを複数用意して，時刻に合わせてリスト間でタイマ要求の付替えを行います。

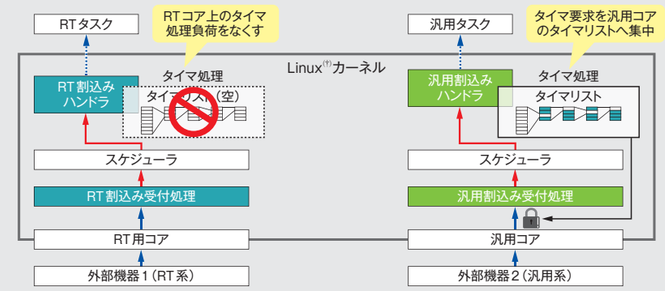

図4．RT用コアのタイマ処理の負荷制御 — タイマ要求の登録を汎用コアのタイマリストに限定することで，RT用コアのタイマ処理の負荷をなくします。

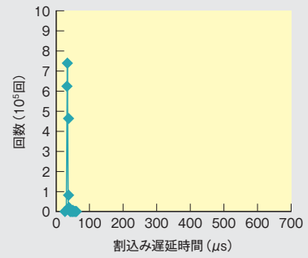

図5．考案した手法を適用した場合の測定結果 — もっとも長い割込み遅延時間が60 μs以下となり，RT性が良いことを示しています。