64ビットプロセッサに向けた実時間保証OSの設計

1F－3

落合 真一†、三上 和敬‡
†三菱電機(株) 情報技術総合研究所
‡三菱電機(株) 電力・産業システム事業所

1. はじめに

産業システムを統括、制御する計算機に対しても、高性能化要求は次のように大きい。

・制御点数の増大要求：デバイス数、コントローラ数
・制御精度の増大要求：ミリ秒制御からμ秒制御へ
・情報処理との融合：インテリジェント化、メディアデータ処理など

特に、制御処理と情報処理の融合は、産業用制御計算機が、汎用の情報処理計算機と同等以上の性能を持つことが要求される。このような産業用制御計算機を実現するために、高性能な 64 ビットプロセッサを採用することを検討している。本稿ではその一ステップとして 64 ビットプロセッサ上で実時間応答性を保証するOSの設計課題と解決策について述べる。

2. 産業用制御計算機における64ビットプロセッサの得失

産業用制御計算機に最新の64ビットプロセッサを適用することは、次のような利点がある。

(1) 性能の向上：アーキテクチャの革新により、従来プロセッサより高い実行性能が期待できる。
(2) レジスタ幅の拡大：数値演算性能が高速化される。
(3) 物理アドレス空間の拡大：大容量のメモリ実装、大きなI/O空間を設定できる。
(4) 論理アドレス空間の拡大：データ配置制約の回避、メディアデータ処理へのメモリマップトファイルの活用などが可能になる。

産業用制御計算機のOSでは、応答性、信頼性、連続動作性などの面から、汎用OSとは異なる思想に基づく設計が必要である。中でも応答性について考えた場合、64 ビットプロセッサ上で OS の実時間性を保証するには新たな課題が生じる。

(1) 割込み処理実行のオーバヘッドの増大
(2) プロセッサコンテキスト量増加によるコンテキストスイッチオーバヘッドの増大
(3) アドレス空間の拡大による管理オーバヘッドの増大

これらの問題により、従来のOS設計を64ビット化するだけでは、プロセッサ性能向上にも関わらず、実時間応答性は悪化すると考える。

3. 64ビットリアルタイムOSの設計

前章で挙げた課題を解決するOS設計を検討した。現時点では、ターゲットのプロセッサを特定していないが、Alpha、UltraSPARC、IA-64などを想定している。

3.1. 割込み処理の単純化

現在の OS 設計では、割込み処理の実時間応答性を高めるために、OS 内処理をフルプリエンプション可能にしている。割込みが入り、高い優先度のスレッドが実行可能になると、現在実行中のスレッドの実行を横取りし、直ちに高い優先度のスレッドにスイッチする。

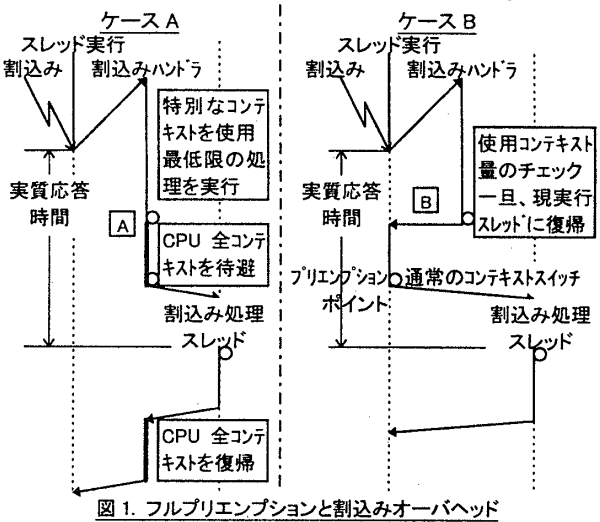

図 1. フルプリエンプションと割込みオーバヘッド

図1ケースAのように、割込みハンドラは特別なコンテキストで動作し、最低限の割込み処理を実行する。実質的な割込み処理は通常のコンテキストで動作するスレッドで行う。この非同期コンテキストスイッチを実現するために、割込みハンドラではプロセッサの全コンテキストを保存し、割込み処理スレッド実行後、その復帰を行う必要がある。しかし、たとえばIA-64では、汎用レジスタ、浮動小数点レジスタとも100本以上もあり、この処理のオーバヘッドは大きい。しかも、コンテキストスイッチ処理はキャッシュヒットを期待できない。

この問題を解決するために、図1ケースBのように、プロセッサの使用コンテキスト量が多い場合には、コンテ

An Examination of Real-time Operating System Design for 64-bit Processors
Shinichi OCHIAI, Kazuyoshi MIKAMI
Mitsubishi Electric Corporation

キストを待避して非同期コンテキストスイッチを行うのではなく、現実行中のスレッドの処理をプリエンプションポイントまで継続し、通常の同期コンテキストスイッチにより割込み処理スレッドを実行する。この2方式の選択により実行性能、応答性能双方の最適化を図る。

3.2. コンテキストスイッチオーバヘッドの削減

同期的なコンテキストスイッチは関数呼出し規約に則って実現している。この場合も、レジスタ幅、レジスタ数の増加がオーバヘッド増大につながる。メモリ性能はプロセッサ性能に比べると、急速な性能向上は見込めないので、応答時間のボトルネックとなる。

この問題を解決するために、コンテキストスイッチ時に大量にあるCPUレジスタを一度に待避、復帰するのではなく、必要となる時までレジスタスイッチを遅らせる遅延コンテキストスイッチを導入する。図2のように汎用レジスタ、浮動小数点レジスタとも群1、群2の2セットに分ける。コンテキストスイッチが必要となると、スレッド動作に最低限必要な群1の汎用レジスタのみを切り替えて、コンテキストスイッチを完了させる。その他の群2の汎用レジスタや浮動小数点レジスタは、スレッドがそれらのレジスタを必要とした時に切り替える。図2の例では、スレッドAからスレッドBへのコンテキストスイッチ時に、まず群1の汎用レジスタのみを切り替え、その他のレジスタはスレッドAのコンテキストのままとする。その後スレッドBの実行により浮動小数点レジスタや群2の汎用レジスタが必要となると、その時に初めてレジスタの遅延コンテキストスイッチを行う。これにより、無駄なレジスタ待避、復帰を減らし、実行時のレジスタ使用量が少ない割込み処理スレッドや、システムスレッドへのスイッチを高速化する。

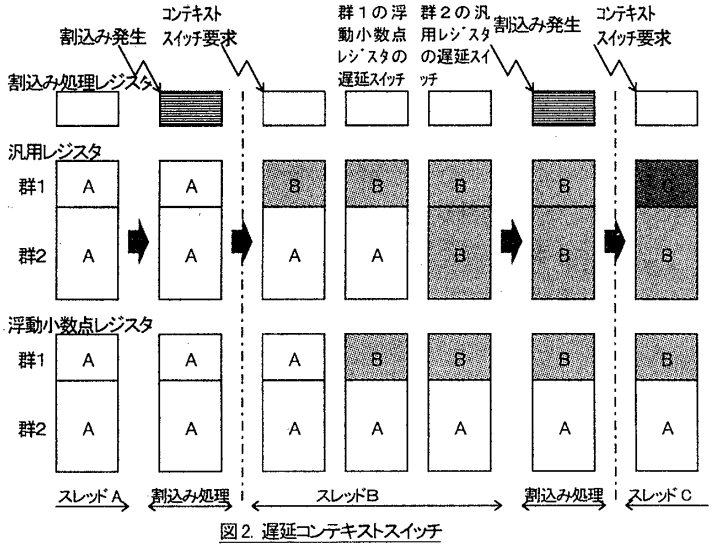

図2. 遅延コンテキストスイッチ

3.3. 論理アドレス空間管理オーバヘッドの削減

64ビットアドレス空間を有効に使うためには、論理アドレス空間上でのシステム動作が必要である。しかし、アドレス空間の拡大により、論理アドレス・物理アドレスの変換オーバヘッドが実時間応答性に無視できなくなる。そこで、図3のような構成のアドレス変換機構を用意する。論理アドレスはまず、ハッシュテーブルを使い検索

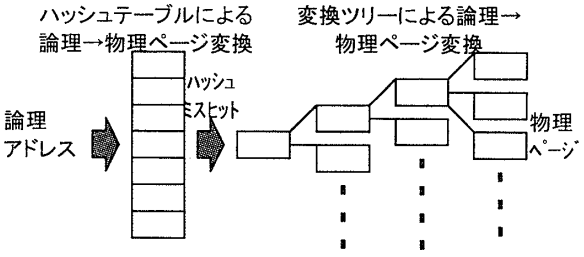

図3.論理アドレス変換オーバヘッドの削減

する。論理アドレスはハッシュにヒットすれば、一回のテーブル参照でアドレス変換が行える。ミスヒットの場合はツリー構造の変換テーブルを検索し、アドレス変換を行う。さらに変換オーバヘッド削減のために、(1)ページサイズの拡大、(2)不要アドレスビットの縮退、(3)ページテーブル領域限定によるポインタサイズ縮退、を行う。ツリー構造テーブルにより、一定回数のテーブル参照でアドレス変換を行う共に、その変換テーブルのリーフ段数を32ビット版と同じにし、アドレス変換の最悪時間保証とオーバヘッド増大防止を両立させる。

4. 性能試算

シミュレーションにより本稿設計の性能試算を行った。

(1) 割込み応答・コンテキストスイッチ応答時間
→ 当初設計版の1/4、32ビット版OSと同等性能
(メモリ性能は4倍を仮定)

(2) 論理アドレス変換オーバヘッド
→ 最悪4段のテーブル検索でアドレス変換を実現、32ビット版OSと同等(変換テーブルのメモリ使用量は32ビット版の2倍)

5.おわりに

本稿では64ビットプロセッサを採用した実時間OSの設計を行った。その結果、従来の32ビット版OSから実時間応答性能を落とさずに、64ビットプロセッサの高性能を享受できる見込みを得た。今後OS試作を行う予定である。

参考文献

[1] M. Brehob, T. Doom, et al., Beyond RISC - The Post-RISC Architecture, Technical Report, Michigan State University of Computer Science