単一仮想記憶型OSにおける外部スケジューラの実装

5L－5

寺本 圭一　岡本 利夫

(株)東芝　研究開発センター　情報・通信システム研究所

1 はじめに

現在我々が開発中のOS(Cubix(CUBe of 2 byte unIX)[1, 2])は，広大な64ビット・アドレス空間を利用した単一仮想記憶空間のモデルを採用している．これにアクセス制御リストに基づくメモリ・セクションと呼ぶページ単位の高機能メモリ保護機能を導入し，プロセス間通信(IPC)をサブルーチン・コールと同様な形式で安全かつ高速に実行するための機構を提供することによって，OSの通信コストを低減させることを目標としている．

本OSで提供しているIPC方式には，EPC(External Procedure Call)とITC(Inter Thread Communication)の2つが用意されており，前者は単一スレッドによるメモリ・セクション間コール，後者は複数スレッド間での通信のために用いられる．

これら各IPC方式における性能評価を行った結果[3]から，EPCでの速度向上はかなり認められたものの，ITCではスレッド切替え時に呼ばれるスケジューラのオーバヘッドが大きく，性能が劣るという問題点があった．

本OSでは，スケジューラをマイクロカーネル方式に基づき，カーネル外部に出すことにより，“カーネルの単純化”と“スケジューリング方式の高度化”の両立を狙っている．初期の外部スケジューラは，その役割に応じて，スレッドの選択や待ち行列操作を行うポリシー部と，選択された次期実行スレッドへの切替えのみを担当するメカニズム部とに分離されていたが，今回改良した試作版外部スケジューラでは，メカニズム実現部として次期実行スレッドの固定的な選択・切替え操作，ポリシー実現部としてプライオリティ制御などによる待ち行列操作，という2階層に分けて実現することにより，スケジューラが外部にあっても高速に動作するための仕組みを提供した．

2 マイクロカーネル構成における外部スケジューラ

本OSでは，外部スケジューラ方式および2段階スケジューリング方式を採用している．これは，スケジューラをカーネル外部に配置することにより，カーネルの変更なしに柔軟にスケジューラ構成を変更するといった自由度が増加するという利点と，スケジューラを高速スケジューラと高機能スケジューラとに分割し処理を緊急度に応じて2段階に処理することによりスケジューリング・ポリシーの明確な分離が可能になるという2点を重視した結果である．

上記のスケジューリング方式を実現するために，初期に試作した外部スケジューラでは，2階層に分かれるスケジューラにそれぞれ以下のような役割を持たせて実装を行っていた(図1)．

1. 高速スケジューラ
   - カーネルより通知される各種イベント(システムコール，割り込み，例外処理など)の受信
   - 受信イベントに対応するイベント・ハンドラへの分岐(サブルーチン・コール)
   - 高機能スケジューラへの呼び出し
   - 次に実行するよう指示されたスレッドへの切替え
2. 高機能スケジューラ
   - スケジューリング・ポリシーに応じて，プライオリティ・キューから次期実行スレッドを選択

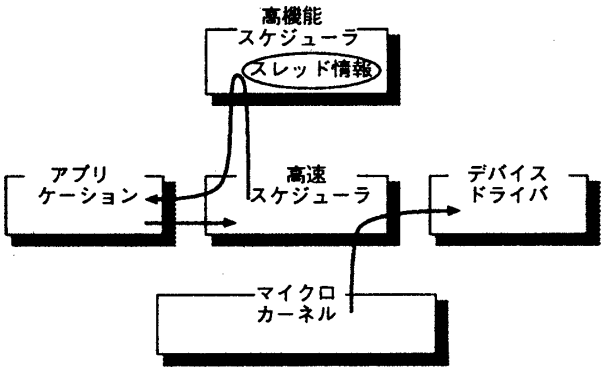

図1: 外部スケジューラ構成概念図(初期)

すなわち，緊急に処理する必要のあるデバイス・ドライバ用のスレッドや最重要スレッドへの切替えを高速スケジューラ(割り込み禁止状態)によって高速に行い，プライオリティ制御やスレッド選択などのスケジューリングに関する処理を高機能スケジューラ部(一部割り込み禁止状態)で分担していた．スレッド情報(待ち行列など)は，高機能スケジューラ内に格納されて操作される．

実際にスケジューラがあるスレッドから別スレッドへ実行スレッドを切替える場合，まず，高速スケジューラから高機能スケジューラに対して次期実行スレッドの選択を依頼する．

An External Scheduler Model for a Single Address Space Operating System

Keiichi TERAMOTO, Toshio OKAMOTO
TOSHIBA CORPOLATION, R&D Center
1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki 210, Japan

続いて，高機能スケジューラは，この要求に応じて選択した実行スレッドをイベントバッファを介して高速スケジューラへ通知する．結果を受け取った高速スケジューラは，その実行スレッドへと制御を移すことになる．

このように，初期の試作外部スケジューラでは，次期実行スレッドの選択の度に，高速・高機能スケジューラ間でコンテキスト切替えが多発するという問題点があった．

3 外部スケジューラの改良

こうしたスレッド選択時の高速・高機能スケジューラ間の通信コストを押えるため，両スケジューラの機能分担を見直し，以下のような改良を行った．

- 固定的に求まるスレッドの選択は高速スケジューラ内で行う
- 高機能スケジューラは，プライオリティ変更操作時や，あるタイムクォンタムに従って高速スケジューラより呼び出され，プライオリティ制御などを行う

すなわち，スレッド選択を固定的に行える範囲では，その選択および切替えまでを高速スケジューラ内で行い，プライオリティの変更などを伴うような範囲にのみ，高機能スケジューラを呼び出す．このため，待ち行列などが含まれるスレッド情報は，高機能・高速スケジューラ間で共有可能にしている．

例えば，アプリケーションからスケジューラを呼び出すシステムコール `call_scheduler` を使用して，プライオリティ変更や sleep/wakeup を実行する際には，最高優先度の待ち行列位置などが変更される可能性があるため，高速スケジューラ内から高機能スケジューラが呼び出される (図 2)．

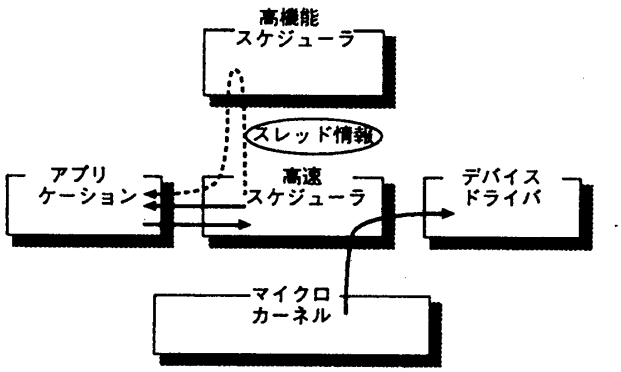

図 2: 外部スケジューラ構成概念図 (改良版)

4 性能測定

ITC によるプロセス間通信を用いるプログラムでは，異なるスレッド同志でコンテキスト切替えを頻繁に行う可能性がある．本 OS における外部スケジューラはこれらスレッド切替えの間に介在してスケジューリングを行うので，外部スケジューラ自身の処理に加え，外部スケジューラとスレッドとの切替え処理をできるだけ短縮することが望まれる．

初期の外部スケジューラでは，1 回の ITC においてユーザスレッドと高速・高機能外部スケジューラ間で合計 4 回のコンテキスト切替えが行われていたが，改良版外部スケジューラを用いることにより，2 回のコンテキスト切替えで処理が完了される．

そこで，コンテキスト切替え回数の低減に伴う ITC 性能向上の度合を調べるために，まず，高速スケジューラとユーザスレッド間の切替えに関連するプリミティブ操作に要する時間を測定した．ここでは，J3100PV(486/66MHz) を用いて，`thread_next`(高速スケジューラ内から選択されたスレッドへと実行を切替えるのに要する時間) と，`call_scheduler`(ユーザスレッドが CPU を放棄するシステムコールを発行してから高速スケジューラを呼び出すまでの時間) について測定し，`thread_next` では 0.052ms，`call_scheduler` では 0.046ms という結果を得た．

次に，1 回の ITC によるスレッド切替え時間を測定し，以下のような結果を得た．

初期版 ITC	改良版 ITC
0.212ms	0.103ms

よって，ITC によるユーザスレッド同志の切替えは，そのほとんどをスケジューラ，ユーザスレッド間のコンテキスト切替えが占めているため，このコンテキスト切替えの回数の最小化は，性能向上に寄与する．

5 おわりに

今回の外部スケジューラの改良では，2 段階スケジューラ方式のもとでスケジューラを，次期実行スレッドの固定的な選択・切替えを行うメカニズム実現部と，プライオリティ制御などのスケジューリングに関するポリシー実現部とに分離することによって，スレッド切替え時にスケジューラが関与するオーバヘッドの低減をはかった．

完全外部スケジューラ化を実現しようとしているため，今回のようなスレッド選択部の改良によってもある程度の性能劣化は免れないが，単一仮想記憶方式に基づき，スケジューラと各種デバイスドライバ・スレッド間とのコンテキスト切替えに EPC による通信方式を取り入れることにより，割り込み処理時のオーバヘッドの削減が可能であると考えている．

参考文献

[1] Okamoto et al., "A Micro Kernel Architecture for Next Generation Processor", pp.83-94, Micro-kernels and Other Kernel Architectures Symposium, USENIX, 1992/4.

[2] 福本 淳 他,「次世代アーキテクチャ向けオペレーティングシステム・マイクロカーネルの開発」, 情報処理学会第 47 回大会, 6B-6, 1993/10.

[3] 津田 悦幸、福本 淳、寺本 圭一、友田 一郎、岡本 利夫,「次世代アーキテクチャ向けオペレーティングシステム: マイクロカーネルの評価」, 情報処理学会 第 49 回全国大会, 1994/9.