並列度の変化を考慮した複数プロセス生成手法

1P−1

横山　和俊　　谷口　秀夫
ＮＴＴデータ通信（株）開発本部

1．はじめに

並列処理を行なう環境においては、オペレーティング・システム（以降、ＯＳと略す）が並列処理単位であるプロセスを効率良く制御する必要がある。プロセス制御機能のうち、プロセス生成処理は、ディスクＩ／Ｏ処理を伴うため処理時間が大きい。特に、並列処理中に必要なプロセス数や実行プログラム（以降、これらをまとめて並列度と呼ぶ）が動的に変化する場合、複数プロセスの生成処理が頻繁に発生することになり、高速なプロセス生成処理が必要になる。プロセス生成処理を高速化する試みとして、ＤＩＲＯＳ（DIstributed Real-time OS）では、複数のプロセスを一括生成するマルチプロセス生成機能を提案している[1]。本稿では、並列度の変化を考慮したマルチプロセス生成機能の高速化手法について述べる。

2．マルチプロセス生成機能

2.1　従来の生成手法

マルチプロセス生成機能の処理内容を示す。

① プログラムをディスクからロードする。
② 最初の１個のプロセス（原本プロセスと呼ぶ）を生成する。
③ 原本プロセスのメモリ上のプロセス生成情報をコピーし、残りのプロセス(コピープロセスと呼ぶ）を生成する。
④ 各プロセスに引数を設定する。
⑤ 走行キューにプロセス群を接続する。

2.2　問題点

並列度が動的に変化する場合、プロセス数の変化が実行プログラムの変化を伴う場合（形態Ａ）とプロセス数の変化が実行プログラムの変化を伴わない場合（形態Ｂ）がある。（形態Ａ）の生成処理については、マルチプロセス生成機能が高速である。（形態Ｂ）の生成処理については、ディスクＩ／Ｏ処理が必要でない生成形式（UNIXのforkシステムコールなど）が有効である。マルチプロセス生成機能は必ずディスクＩ／Ｏ処理を伴うため、（形態Ｂ）の生成処理については、低速である[2]。

3．高速化手法

（形態Ｂ）の生成処理においてディスクＩ／Ｏ処理が発生する問題点を解決するために、「プロセス生成に利用したディスク格納情報をメモリ上に保持する方式」を提案する。

3.1　高速化の手段

高速化を実現するためには、利用するメモリ情報の形態とその利用方式を決定することがある。

(1) 利用可能なメモリ情報

以下の３つの形式がある。

(A) ディスク格納情報をメモリ上に保持（常駐）し、プロセス生成の時に利用する形式。保持する情報の形態として、以下の２つがある。
　(a) テキスト部とデータ部
　(b) プロセスイメージ（テキスト部、データ部、スタック部、および管理テーブル）
(B) あらかじめ、走行直前のプロセスを多数生成しておき、プロセス生成の時に利用する形式。
(C) (A)と(B)の組合わせ

(2) 利用方式

メモリ情報を利用する方式は、プロセス生成処理の開始点と関係が深い。プロセス生成処理の開始点として、以下のものがある。
　(a) システムコール発行直後（①の前）
　(b) ディスクＩ／Ｏ処理の直後（①の後）
　(c) 原本プロセス生成の直後（②の後）
　(d) コピープロセス生成の直後（③の後）

(3) 具体的な高速化方式

(1)と(2)の組合わせを考えると、表１に示す５種類の高速化方式が考えられる。以下に各々について説明する。

＜方式１、方式２＞

ディスク格納情報をメモリに保持し、高速化を実現する。１回目のプロセス生成の時に、メモリ上にディスク格納情報を保持する。２回目以降のプロセス生成では、メモリ上の情報を利用しディスクＩ／

A Multi-process Creation Mechanism on Concurrent Processing
Kazutoshi YOKOYAMA and Hideo TANIGUCHI
NTT DATA COMMUNICATION SYSTEMS CORPORATION

表１　高速化手法

方式	メモリ上の情報	２回目の生成の開始点
方式１	テキスト部 データ部	原本プロセス生成
方式２	原本プロセス	コピープロセス生成
方式３	コピープロセス	引数の設定 または ディスクＩ／Ｏ
方式４	テキスト部 データ部 コピープロセス	引数の設定 または 原本プロセス生成
方式５	原本プロセス コピープロセス	引数の設定 または コピープロセス生成

Ｏ回数を削減させる。

＜方式３＞

あらかじめ走行状態直前のプロセスを生成し利用する形式による高速化手法である。具体的には、１回目のプロセス生成のときに、③の処理で余分にコピープロセスを生成する。コピープロセスが不足した場合、ディスクＩ／Ｏ処理から開始し、１回目と同じ処理を行なう。

＜方式４、方式５＞

メモリ上に保持された情報を利用するし、さらにコピープロセスを利用する。

３．２　性能比較と考察

各方式の性能比較図を図１～図３に示す。

(1) １回目の生成処理

図１に示すように現行方式が最も高速であり、次いで方式１～方式５の順である。また、プログラムサイズが増加するに伴い、処理時間の差が大きくなる。これは、２回目以降の生成処理を高速化するため、メモリに必要な情報を保持する処理が増加するためである。

(2) ２回目以降の生成処理

図２に示すように、ディスクＩ／Ｏ処理の回数を削減するには、テキスト部とデータ部の保持が有効である。さらに、コピープロセスを持つ方式はプログラムサイズに影響を受けず、最も高速である。

しかしながら、図３に示すように、生成するプロセス数が保持しているコピープロセス数を越えた場合、１回目の生成処理と同じ処理が必要となり、コピープロセスを持たない方式（方式１や方式２）より低速になる。

(3) 性能比較のまとめ

コピープロセスを利用する方式は、使用するメモリ量が多量であるにも関わらず、高速化はわずかである。また、プログラムサイズが大きく、生成プロセス数が多いときは低速になることが考えられる。したがって、方式１と方式２が性能とメモリ量の両方に対して有効であると考えられる。

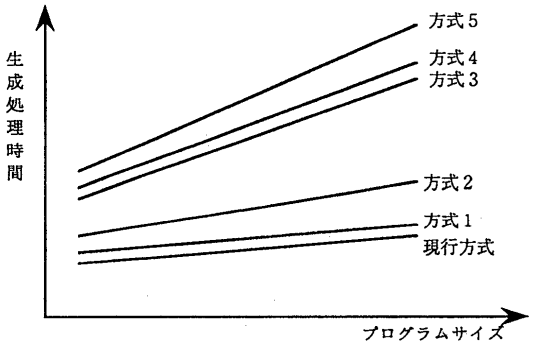

図１　１回目のプロセス生成処理時間の比較

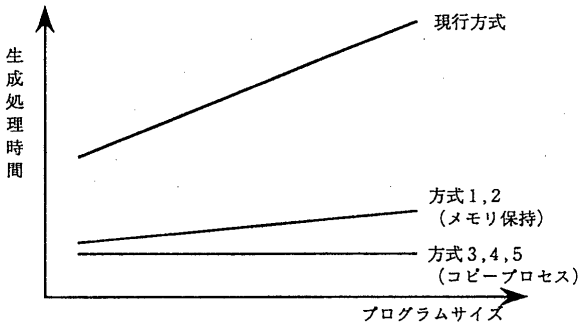

図２　プロセス生成処理の高速化の比較

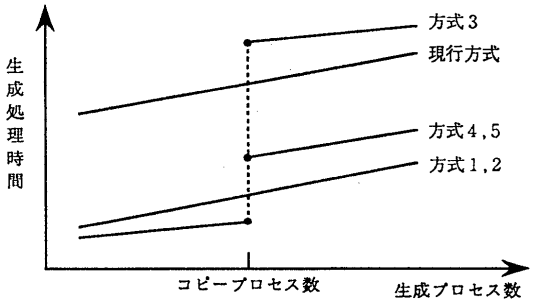

図３　２回目のプロセス生成処理時間の比較

４．おわりに

マルチプロセス生成機能の高速化手法について述べた。今後は、定量的な評価を行なう予定である。

＜文献＞

[1] 谷口，他："プロセス生成の高速化と並列化に関する検討"，信学報，CPSY89-28 (1989).
[2] 横山，他:"多数プロセス生成の高速化手法"，SWoPP'92 発表予定 (1992).