分散環境におけるプロセス間依存関係の利用に関する検討

4F－5

田辺 雅則　伊藤 健一　谷口 秀夫
NTTデータ通信(株)　NTTデータ通信(株)　九州大学 工学部

1 はじめに

サービスを複数のプロセスで構成し、複数のプロセッサに分散させて実行させる分散処理では、プロセスの配置方法の違いによってプロセッサ間の通信量が変化するために処理の効率が変わる。

プロセス間の依存性のうち、同期と通信の依存性をプログラムから抽出する研究が行われている[1][2]。これらの依存をもとにして分散配置を行うことで、並行処理が行えるプロセスを抽出できる。しかし、効率の良い分散配置を行うためには、資源競合や処理の順序制約による相手プロセスの処理待ちなどの問題解決が必要である。

本報告では、プロセス間の資源競合と順序制約の依存関係を利用して問題解決を行い、効率の良いプロセス配置を行う方法の検討を行った。

2 依存関係

2.1 分散配置に必要な依存関係

プロセス間の依存関係を以下に示し、2.1.1から2.1.3節で説明を行う。

順序制約　：優先順位、処理手順
資源競合　：排他制御、I/O処理
データ通信：同期、データ送信/受信

2.1.1 順序制約

(1) 優先順位

プロセスはすべて同じ優先順位でプロセッサにスケジューリングされるのではなく、処理内容によって異なっている。例えば、データ入力や画面制御などを行うプロセスは、優先順位を高くすることが考えられる。

(2) 処理手順

プロセスの処理が終了した後に別のプロセスの処理を行う処理手順の制約がプロセスにある。例えば、トランザクション処理においては、データベースを更新する場合はログをとってからデータベースを更新するという処理手順の制約がある。

2.1.2 資源競合

(1) 排他制御

複数のプロセスが資源を共有する場合、排他による資源競合が発生する。競合したプロセスは先に資源を確保したプロセスが資源を解放するまでwait状態になるため、他に走行させるプロセスがないとプロセッサの使用効率が悪くなる。

(2) I/O処理

ディスクや通信路などの資源を使用する場合、I/O処理が発生する。通常、1つの資源に対して、同時に実行できるI/O処理は限られる。そのため、同時に複数のプロセスが1つの資源を利用する場合、先にI/O処理を行ったプロセスがその処理を終了するまで他のプロセスはwait状態になる。その場合、他に走行させるプロセスがないとプロセッサの使用効率が悪くなる。

2.1.3 データ通信

(1) 同期

複数のプロセスが協調して処理を行う場合、プロセス間には同期が起こる。先に相手プロセスの同期をもとめたプロセスは、wait状態になるため、他に走行させるプロセスがないとプロセッサの使用効率が悪くなる。

(2) 送信/受信

データを送信/受信する場合、通信路の遅延やデータ転送処理に時間がかかる。また、プロセスはそのデータを送信/受信するためにwait状態になる。その時に走行させるプロセスがないと、プロセッサの使用効率が悪くなる。

以上のような依存関係を使うことで、プロセス配置が効率的にでき、処理時間を短くさせることができると考えられる。

2.2 依存関係を含むモデルの仮定

2.1節で述べた3つの依存関係を含むモデルを仮定する。資源には、データベースを考える。この処理モデルは、表1に示すようにAからE_2の6個のプロセスと1から3の3個のデータベースで構成される。

このモデルに入力されたデータは、Aから始まる6個のプロセスによって順次処理されたあと、結果として出力データが返される。

プロセスの処理時間の仮定を以下に示す。データベースへのアクセスは、プロセスの実行開始から25ミリ秒後に、50ミリ秒連続して行われることにした。

(1)プロセス処理　：100ミリ秒
(2)プロセッサ間通信　：50ミリ秒
(3)データベースアクセス：50ミリ秒

Programming Language for Distributed processing
†Masanori TANABE, †Kenichi ITOH & ††Hideo TANIGUCHI
†NTTDATA COMMUNICATION SYSTEMS CORPORATION
††KYUSYU UNIVERSITY

表 1: 仮定した処理モデルを構成するプロセス

プロセス	他のプロセスとの依存関係	データベース番号	優先順位
A	B： データ通信 (送信)	未使用	2
	D： データ通信 (送信)		
B	A： データ通信 (受信)	1	2
	C： データ通信 (送信)		
C	B： データ通信 (受信)	2	2
	A： データ通信 (送信)		
D	A ： データ通信 (受信)	1	1
	E_1： データ通信 (送信)		
	E_2： データ通信 (送信)		
E_1	A ： データ通信 (送信)	3	1
	D ： データ通信 (受信)		
	E_2： 順序制約 : (E_2 より先に実行する)		
E_2	A ： データ通信 (送信)	3	1
	E_1： 順序制約 : (E_1 のあとに実行する)		

注. 優先順位： 数字の小さい方が優先順位は高い。

3 依存関係を利用することの効果

3.1 依存関係を考慮しないプロセス配置

(1) プロセス配置

依存関係を考慮しないプロセス配置を表 2に示す。プロセスの生成順に A をプロセッサ 1 に、B をプロセッサ 2 へと配置した。データベースについては、1 つのプロセッサに 1 つとした。

表 2: プロセスとデータベースの配置

プロセッサ番号	プロセス		データベース
1	A	D	1
2	B	E_1	2
3	C	E_2	3

プロセスが実行される処理順序を図 1に示す。全体の処理時間は、1000 ミリ秒となる。

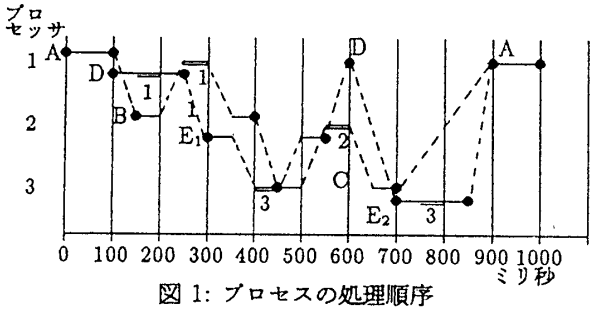

図 1: プロセスの処理順序

実線 ：プロセスのプロセッサ使用時間
二重線 ：データベースへのアクセス時間
点線 ：プロセッサ間をまたぐデータ通信
アルファベット：プロセス名
数字 ：使用するデータベース番号

(2) 考慮する依存関係

(1) のプロセス配置の結果より、以下の 2 つの依存関係に着目した。

順序制約： プロセスは、順番に処理されなければならないという制約のため、すべてのプロセッサに空き状態が生じている (実線の記入されていない部分が空き状態を示している)。

データ通信： 異なるプロセッサ間でのデータ通信が少なくなるようなプロセス配置でないため、通信時間がかかっている。

3.2 依存関係を考慮したプロセス配置

以下の規則にもとづき、プロセス配置を行った結果を表 3に示す。A はデータベースを使用しないが、プロセッサにプロセスが均等になるように配置した。

規則： データベースの配置場所とプロセスの配置場所の違いにより増えている通信時間を減らすために、プロセスが使用するデータベースと同じプロセッサにプロセスを配置する。

表 3: 依存関係を考慮した配置

プロセッサ番号	プロセス		データベース
1	B	D	1
2	A	C	2
3	E_1	E_2	3

依存関係を考慮した配置における処理の流れを図 2に示す。結果、処理時間が 900 ミリ秒となり、考慮しない場合より効果があることが確認できた。

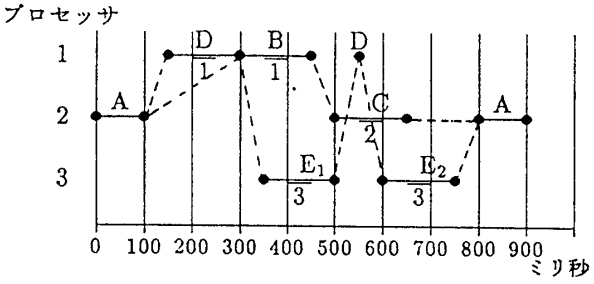

図 2: 新しい配置での処理順序

4 まとめ

プロセス配置にデータ通信の依存関係を利用することの効果を 1 つのデータの処理について机上検討した。今回仮定した条件のもとでは、依存関係を利用することにより、有効性が得られることが分かった。

今後の課題として以下のことがあげられる。

(1) 実際にプロセスを配置しその有効性の検証
(2) 複数のデータが連続到着する場合の配置問題

参考文献

[1] Karl J. Ottenstein, 他 "The Program Dependence Graph in a Software Development Environment", ACM Software Engineering Notes Vol.9 No.3 1984.5

[2] 笠原 他 "並行プログラムのためのプロセス依存ネット生成ツール", 情処研報 PRG 13-5, 1993.8.20