分散環境上での並列プログラミング言語処理系用のOS DaOSの概要

6－5 9

岡村耕二＊　縄田毅史＊　平原正樹＊＊　荒木啓二郎＊
＊九州大学 工学部　＊＊九州大学 中央計数施設

1. はじめに

我々は並列プログラミング言語の処理系を分散環境上で実現する研究を行っている。本論文ではこの処理系の言語仕様で記述された並列プログラムの実行を支援するオペレーティングシステム(以下OS)、DaOSの概要を述べる。

本論文では処理単位の呼び方を扱い方によってプロセスとタスクとに区別する。すなわち、処理単位をOSで管理する場合はプロセスと呼び、処理単位を言語のレベルで指定する時はタスクと呼ぶ。

2. DaOSのあらまし

DaOSはイーサネットで接続されたUNIXワークステーションによる疎結合分散環境上で、並列プログラミング言語処理系ParaDisE[1]の言語仕様で記述された並列プログラムの実行を支援するOSである。

DaOSはこのような疎結合分散環境上で、並列プログラムとのインタフェースになるランデブなどの並列プログラミング言語特有の機能を提供することと、ParaDisEとのインタフェースにおいて前提としている単一アドレス空間モデル[2]を実現することにより、並列プログラムを分散環境上で実行することを支援している。

DaOSの開発では、並列プログラミング言語を支援するサービスや分散環境に対応するサービス等の新しい技術に重点を置き、既存の技術で利用できるものはそのまま利用した。そのため、DaOSはワークステーションのUNIX上に構築し、メモリ管理、入出力管理等のハードウエアとのインタフェースの多くは、UNIXに任せた。

3. DaOSの基本的な機能

DaOSの基本的な機能を、メモリ管理、プロセス管理、入出力管理およびネットワーク管理に分けて説明する。

3.1 メモリ管理

メモリ管理で必要なのは、i)プロセスの各種テーブルの確保、ii)動的なメモリ領域の確保である。

プロセスの各種テーブルはDaOS立ち上げ時に静的に確保する。スタック領域もこれに含まれる。動的なメモリ領域の確保にはUNIXの関数mallocを用いている。

mallocに失敗した時、そのプロセスは消滅する。同様に、スタック領域を使い果たしたプロセスはその時点で消滅する。このようにメモリ管理はUNIXに依存している。

3.2 プロセス管理

DaOSは単一ワークステーション上ではUNIXの1つのプロセスを時分割してマルチプロセスを実現している。プロセス管理には教育用に開発されたOS、XINU[3]の一部を用いた。

プロセスの状態には、i)実行可能(READY)、ii)実行中(CURRENT)、iii)停止(SUSPEND)、およびiv)待機(WAIT)がある。

プロセスIDは各ワークステーション上で局所的に付けられる。プロセスの切替えは50msecの間隔で行い、優先度の高いプロセスから実行する。

DaOS:Operating System for Parallel Programming Language in Distributed Environment
Koji OKAMURA*, Takeshi NAWATA*,
Masaki HIRABARU** and Keijiro ARAKI*
*Faculty of Engineering, Kyushu University
**Computation Center, Kyushu University

3.3 入出力管理

DaOSの入出力には、i)画面出力、ii)キーボード入力、iii)ファイルの入出力および、iv)ソケットの入出力がある。これらの入出力にはUNIXの関数(printf、scanf)や、システムコール(read、write)をそのまま用いている。

ii)キーボード入力と、iv)ソケット入力には本来割り込みを用いるべきであるが、割り込みで入力を検知するためには別のUNIXのプロセスを必要とする。それはDaOSの内部構造を複雑にするため、同一のUNIXプロセスで入力を検知できるようにポーリングを用いた。すなわち、入力を処理するプロセスは普段は待機状態であり、DaOSは50msec間隔で起こるタイマ割り込み時に入力がなされていないか調べ、入力されていれば、その入力を処理するプロセスを実行可能状態にする。

3.4 ネットワーク管理

DaOSのネットワーク管理のサービスは、分散環境上で単一アドレス空間を実現するためのものが中心で、i)他のワークステーションのアドレスからの読み出し(rac_read)、ii)他のワークステーションのアドレスへの書き込み(rac_write)、iii)非ブロック型遠隔手続き呼び出し(rac_proc_n)、iv)ブロック型遠隔手続き呼び出し(rac_proc)および、v)他のワークステーション上へのプロセスの生成(rac_task)を用意している。

ネットワーク管理はUNIXのソケットを用いた。それぞれのソケットに一つのサーバプロセスがあり、ネットワークのサービスはそのサーバプロセスを通して行われる。

単一アドレス空間のアドレスは、ワークステーション番号と、ワークステーション内アドレスで定義し、タスクIDをワークステーション番号とワークステーション内プロセスIDで定義することによりタスクの指定を分散環境上で一意にしている。

4. 並列プログラミング言語とのインタフェース

ParaDisEの言語仕様におけるタスク機能は並列プログラミング言語Adaに準じてるので、ParaDisEの言語仕様で記述された並列プログラムとのインタフェースをとるためにDaOSはタスク間の親子関係を形成する機能と、タスク間のランデブの機能を用意している。また、並列プログラムから分散環境を隠蔽するグローバルなタスクIDの管理のサービスを提供する。

4.1 タスクの生成と親子関係

親タスクの本体の開始は、自分が生成した子タスクの本体の実行が開始するまで待つ。そして、親タスクの実行終了は自分が生成した子タスクの実行が終了するまで待つ。

単一ワークステーション上でのタスクの生成と親子関係に関する機能は以下の通りである。

タスクの生成
Task_Init(エントリアドレス, 優先度, 引数のアドレス,
　引数のサイズ, アクセプト文の数)
返り値はプロセスID

タスク本体の実行開始
Task_Begin();

タスク本体の実行終了
Task_End();

4.2 ランデブ

タスク間通信はランデブを用いる。 ランデブの引数は1つで, それはアドレス型である。 そのアドレスを使った間接指定法で、 Adaにおけるパラメタ渡しの3種類のモードを持つ多様の通信をすることができる。

単一ワークステーション上でのタスク間ランデブに関する機能は以下のものがある。

エントリコール
Entry_Call(プロセス ID, エントリ ID, 引数のアドレス)

単純なアクセプト
Entry_Accept(エントリ ID, 引数のアドレス)
.....
Entry_Return(エントリ ID)

選択的なアクセプト
```
Select_Accept(エントリ ID_A);
Select_Accept(エントリ ID_B);
switch(Select_Switch()){
case 0:
  Entry_Accept(エントリ ID_A, 引数のアドレス)
  .....
  Entry_Return(エントリ ID_A);
  break;
case 1:
  Entry_Accept(エントリ ID_B, 引数のアドレス)
  .....
  Entry_Return(エントリ ID_B);
  break;
}
```

4.3 分散環境上でのタスク管理

4.1、 4.2 は単一ワークステーション上のタスク間で使用される機能なので、これらを異なるワークステーション上に存在するタスクに対して用いる時は、遠隔手続き呼び出しを使用する。 異なるワークステーション上へのタスクの生成と異なるワークステーション上に存在するタスク間のランデブはしばしば行われるので、これらの機能と遠隔手続き呼び出しを組み合わせて一般的にした機能を用意した。

分散環境上のタスク ID は、 ワークステーション番号とワークステーション内プロセス ID で定義した。しかし、 プログラム中ではワークステーション番号とワークステーション内プロセス ID を指定するより、一次元的に指定できた方が便利である。 この分散環境上で一次元的な ID をグローバルタスク ID と呼ぶ。以下に示す機能ではタスクをグローバルタスクとして扱う。 DaOS がこのグローバルタスク ID の管理をする。

遠隔的にタスクを生成する
trans_Task_Init(ワークステーション番号, エントリアドレス,
  優先度, 引数へのアドレス, 引数のサイズ, アクセプト文の数)
返り値は**グローバルな**タスク ID

グローバルタスク ID に基づくエントリコール
trans_Entry_Call(**グローバルな**タスク ID, エントリ ID,
  引数のアドレス)

Entry_Accept の返り値はエントリコールをかけたタスクの存在するワークステーションの番号であるから、 引数のアドレスと併せれば、 単一アドレス空間のアドレスになる。 したがって、 データ通信を行うためには、 遠隔読み出しや書き込み (rac_read、 rac_write) を併せて用いればよい。

これらの機能は単一ワークステーション上でも、 使うことができる。 結局、 このグローバルなタスク ID を用いれば、 並列プログラムから分散環境を隠蔽し、並列プログラミング言語のレベルでは、 タスクの指定時にそのタスクがどこのワークステーションにあるのかに注意する必要がなくなる。

図1に DaOS の構造を示す。

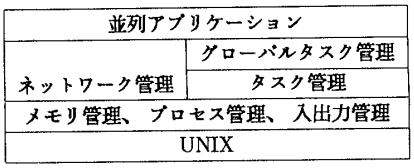

図 1: DaOS の構造

5. 性能評価と今後の課題

我々は予備実験として C 言語で記述した並列プログラムを UNIX ワークステーションによる分散環境上で実行し、 並列計算実行時の UNIX ワークステーションによる分散環境の効率を確認している[4][5]。 この UNIX と C 言語のみで記述した並列計算の結果と DaOS を用いた時の結果を比較して DaOS の性能評価をする。

今後は、 次に述べるような色々な面からの性能評価をしながら、 DaOS のチューニングアップをしていきたい。

5.1 逐次処理

DaOS のマルチプロセスは UNIX のプロセスを時分割して実現おり、 そのために 50msec の周期でタイマ割り込みが起こっている。 そのオーバーヘッドのため、逐次処理でも処理速度の低下がある。

5.2 ネットワーク間サービスに要する時間

DaOS はソケット入力をポーリングで検知しており、 それによる遅延から来る処理速度の低下が予想できる。

また、 ネットワーク間の通信は全てサーバタスクを通して行われるため、 このオーバーヘッドの影響も受ける。

5.3 応用プログラムの実行

最終的には、 多種多様な応用並列プログラムの実行結果より、 総合的な評価を行う。

6. おわりに

DaOS は OS といってもメモリ管理や、 入出力管理等のハードウェアに依存する部分は全て UNIX に任せているため、 本当の OS とは言い難い面もある。 しかし「データ処理を行う際に必要になるさまざまな資源の無駄な遊びをできるだけ排除し、 データ処理の生産性を向上させるねらいのもとに、 各種プログラムを体系的に統合化したもの」[6] という OS の定義に従えば、 DaOS は技術の発達により安くなったワークステーションの CPU 資源を有効に利用している OS ということができる。

謝辞

日頃から御指導して下さる九州大学工学部情報工学科の牛島和夫教授や吉田紀彦助手と、 有益な議論をしてくれる計算機研究室の皆さんに感謝致します。

参考文献

[1] 縄田、岡村、平原、荒木:“並列 / 分散環境上のプログラミング言語処理系 ParaDisE”, 情報処理学会 第 40 回 (平成 2 年前期) 全国大会, 1990 年 3 月

[2] 岡村、縄田、平原、荒木:“単一アドレス空間モデルに基づいた分散環境上での並列プログラミング言語処理系の実現”, 信学技報, Vol.89, No.166, pp.33-38, 1989 年 8 月

[3] D.Comer:“Operating System Design, the XINU Approach,” *Prentice-Hall Software series*, Prentice-Hall, 1984.

[4] 岡村、平原、荒木:“UNIX ワークステーションによる分散環境上での並列数値計算に関する実験と評価”, 平成元年度電気関係学会九州支部連合会大会, 1989 年 10 月

[5] 岡村、平原、荒木:“UNIX ワークステーションによる分散環境上での並列計算に関する実験”, 第 14 回 jus シンポジウム, 1989 年 11 月

[6] 江村潤朗:“オペレーティングシステム入門”, オーム社, 1978 年