Transputerネットワーク上の記号処理用分散カーネル

A Distributed Kernel for Symbolic Languages on the Transputer Network

竹内 幹雄† 岩崎 英哉† 武市 正人†
Mikio TAKEUCHI Hideya IWASAKI Masato TAKEICHI

†東京大学 工学部
Faculty of Engineering, University of Tokyo

概 要

記号処理言語の処理系をTransputerネットワーク上に実現するにあたり，位置透過 (location transparent) な通信を提供し，同時にヒープ領域の管理を行う記号処理言語用分散カーネルを作成した．保守拡張性に優れ，分散環境に適したスレッド指向構成法を採用している．また余分なオーバヘッドの発生を防ぐため，ハードウェアの提供する機能を積極的に利用した．本発表では，これを用いてCommitted Choice型言語Flengの処理系を実現し，カーネルの設計を評価した結果を報告する．

1 はじめに

1.1 研究の目的

本研究は次の2つの機能を持つ分散カーネルをTransputerネットワーク上に構築することを目的とする．

- Transputerネットワーク上でのプログラム開発環境
- 疎結合並列計算機上での記号処理言語の処理系の構成法の研究や評価に用いる実験システム

開発環境の側面では，位置透過な通信を提供した．

位置透過性は分散OSが提供する最も基本的な性質である [1]．ここではカーネルによるメッセージ転送を実現し，スレッドが実行されるプロセッサをユーザに意識させない柔軟なスレッド間通信を提供した．

記号処理の側面では，カーネルにヒープ管理機能を持たせた．従来各々処理系側に実装されてきたごみ集め (garbage collection) を，様々な記号処理言語で利用できるよう汎用化してカーネルに組み込み，処理系作成者の負担軽減を図った．また疎結合並列計算機に特徴的な遠隔参照 (remote reference) の管理もカーネルに組み込んだ．

1.2 Transputer

ここでは疎結合並列計算機の例としてTransputer [2]を用いた．Transputerは以下のような特徴を持つ並列処理用プロセッサである．

- 別のTransputerとの通信に用いるLinkインターフェイスを4つ内蔵
- 2段階の優先度からなるスケジューラを内蔵
- チャネルを用いた同期通信が利用可能

Linkインターフェイス間をケーブルで接続するだけで簡単に疎結合並列計算機が構成できるため，ネットワークトポロジーの変更が容易であり実験システムに適している．

スケジューリングは優先度毎に異なる2つのキューを用いて，次の戦略で行われる．

高優先度 低優先度のスレッドに優先して実行される．チャネルでのブロックを除いてプロセッ

サを横取りされない．

低優先度 高優先度のスレッドにプロセッサを横取りされる．2msのタイムスライスによるラウンドロビン式スケジューリングがなされる．

横取りが起こり得る命令を限定し，退避する情報を減らすことにより，効率的なコンテクストスイッチが実現されている．

チャネルの入出力待ちによるブロックは優先度に関わらずキューから外されるので，ビジーウェイトによる無駄なプロセッサの消費は起こらない．

2 カーネルの概要

本カーネルは以下の機能を実現し，ユーザに対して標準的な手続きを提供する．

2.1 スレッド管理

スレッドとはプログラムにおける制御の流れを抽象化したものである．以下ではカーネル，ユーザプログラムに属するスレッドをそれぞれカーネルスレッド，ユーザスレッドと呼ぶ．

各ユーザスレッドに対して，生成時に全ネットワークで一意に定まるスレッド識別子を割り当てる．それはプロセッサ識別子とプロセッサ内で一意に定まる整数からなり，スレッド間通信やスレッド情報の獲得に用いる．

スレッドのスケジューリングはハードウェアに内蔵されたスケジューラに任せ，余分なオーバヘッドの発生を防いでいる．

2.2 スレッド間通信

スレッド識別子を相手に指定して通信を行う．スレッドが実行されるプロセッサやプロセッサ間の接続関係をユーザが意識する必要のない，位置透過な通信を提供する．通信の種類としては1対1の非同期通信のみを提供し，その他の通信にはライブラリを用いる．

2.3 ヒープ管理

疎結合並列計算機に記号処理言語の処理系を実装する場合，異なるプロセッサのヒープを間接的に参照するために遠隔参照を用いる [3] [4]．遠隔参照の作成，更新にはヒープを公開する側と参照する側の協調が必要であるが，この作業をカーネルが行い，それを利用する手続きをユーザに提供することで，複雑な詳細を理解する必要をなくした．

またごみ集めは他のプロセッサから参照されているオブジェクトも保護しなくてはならないため，単一プロセッサの場合より繁雑な処理が必要となる．これもカーネルに組み込んで，処理系作成者がごみ集めを意識する必要をなくした．

2.4 入出力処理

ファイルシステムへのアクセスを伴う入出力は，ホスト計算機と接続された特定のプロセッサ (ルートプロセッサ) でしか実行できない．そこでルートプロセッサ上のカーネルは入出力のサービスも行う．実際には入出力ライブラリが要求をカーネルに対して送り，カーネルがルートプロセッサにその要求を転送するので，ユーザはスレッドの位置に関係なく同一の手続きを用いることができる．

3 カーネルの実現

3.1 スレッド指向構成法

本カーネルは特定の機能を実行する複数のスレッドがメッセージ交換によって動作するスレッド指向構成法を採用した．実際には以下のスレッドに分割されている (図1)．

- スレッド管理 (Thread Manager)
- メッセージ管理 (Message Manager)
- メモリ管理 (Memory Manager)
- ユーザスレッドからのメッセージ受信 (Thread Receiver)
- ユーザスレッドへのメッセージ送信 (Thread Sender)
- リンクからのメッセージ受信 (Link Receiver)
- リンクへのメッセージ送信 (Link Sender)

カーネルを複数のスレッドに分割することで，新しい機能の追加や既にある機能の拡張が容易になった．また機能別に分割することにより，機能の分散化を実現できた．

カーネルをスレッド指向にした場合に生じる一般的な問題点として，スレッド間の同期，通信に伴うオーバヘッドが大きいことが挙げられるが，Transputer ではチャネルを用いた通信がハードウェアで実現されており，またコンテクストスイッチが十分速いため，生じるオーバヘッドを小さく抑えること

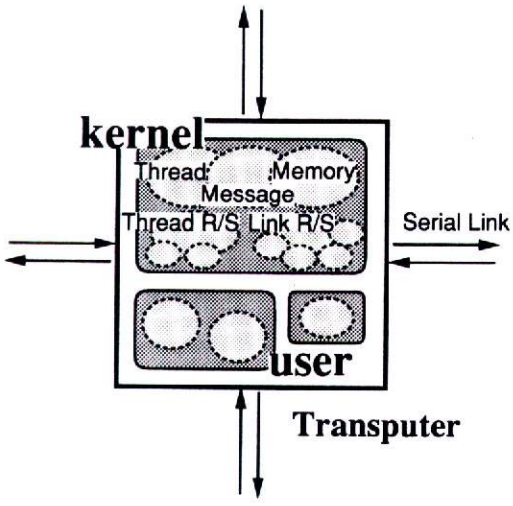

図1　システムの構成

ができた．

3.2 実行モード

Transputerには通常カーネルを実行する際に用いる特権モードと，それに移行するためのトラップ命令が存在しない．しかしチャネルの入出力待ちではどのスレッドもキューから外されることと，優先度によって異なるスケジューリングを利用して，以下の対応により同等の機能を実現できる．

	一般プロセッサ	Transputer
カーネル実行	特権モード	高優先度
カーネル移行	トラップ命令	チャネル出力

すなわちカーネルスレッドは高優先度で，ユーザスレッドは低優先度で実行し，カーネルは仕事の無いときには常にチャネルからの入力を待てばよい．

4 ヒープ管理

4.1 ごみ集め

一般にごみ集めを実行するにはオブジェクトの構造を知っていなければならない．そこで本カーネルでは以下の記号処理言語で一般的に用いられるオブジェクトに関して，カーネルがその構造を規定した．

- Symbol
- Cons
- Vector

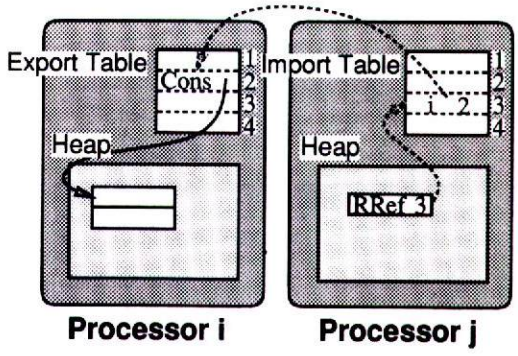

図2　遠隔参照

- String
- Integer
- Reference
- Remote Reference

言語独自のオブジェクトを定義する方法は2つある．1つはオブジェクトの構造を次の3つから選び，オブジェクトタグと組にしてカーネルに宣言する方法である．

- データセル
- 連続する固定個のセルへのポインタ
- Vectorと同一の構造

もう1つはChunkという汎用データ構造を用いる方法である．

局所的なごみ集めにはMorris [5]のアルゴリズムを用いた．

4.2 遠隔参照

遠隔参照とは疎結合並列計算機上で異なるプロセッサのヒープを間接的に参照する手段である(図2)．オブジェクトを他のプロセッサに輸出する際には，輸出テーブルを介して間接的に公開する．それによりごみ集めを局所的に実行することが可能になる．輸入する側は輸入テーブルを介して間接的に参照する．それにより輸入したプロセッサからもはや遠隔参照されなくなったオブジェクトを回収することが可能になる．またオブジェクトの具体化の際には，輸出した側にもその結果を伝えなくてはならない．

本カーネルではオブジェクトの輸出，輸入，具体化に対する標準的な手続きを提供している．

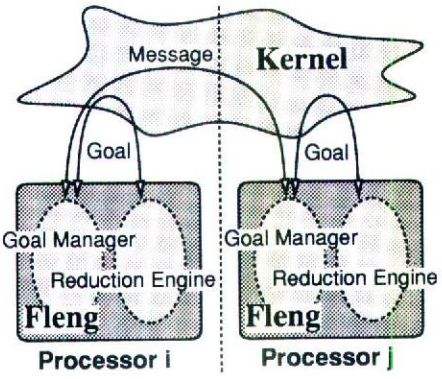

図3 Fleng処理系

5 Committed Choice 型言語 Fleng

本カーネルの提供する機能を用いて動作する記号処理言語の例としてCommitted Choice型言語Fleng [6]の処理系を実装した.

処理系はプロセッサ毎にゴールの管理を行うスレッド(ゴールマネージャ)と簡約を実行するスレッド(リダクションエンジン)の2つからなる(図3). 前者は他のプロセッサとゴールをやりとりして負荷を分散し, 後者はプロセッサの数を意識することなく簡約を行う.

負荷分散には, プログラムに並列実行のための情報を加えて静的にゴールの分配を行う方法と, 実行時にプロセッサの負荷に応じてゴールを動的にやりとりする方法がある. ここではプログラマの負担を減らし, ネットワークトポロジーの変更に柔軟に対応するため, 後者を採用した. また簡単のため, あるプロセッサ上で実行するゴールが1つもなくなったとき, 初めて他のプロセッサ上のゴールマネージャにゴールを要求する方法をとった.

定義節は読み込んだ段階で, あらかじめ全てのプロセッサに送る方法をとった. したがってそのプロセッサでの実行に必要のない節も保持することがあるが, 実行時に節を輸出入することを省略し, 単純な処理を実現した.

6 評 価

オブジェクト指向分散OSにはごみ集めをカーネルに実装したものもある[7]. 本カーネルはシステムをオブジェクト指向で構成するのではなく, 各種記号処理言語で用いられるオブジェクトの共通性に着目し, 汎用のごみ集めを実現することができた.

Fleng処理系の作成に際しては, 言語独自のデータ構造をカーネルが用意したものを用いて容易に定義できた. またカーネルが定めたオブジェクトと同様に, ごみ集めを意識することなく使用することができた.

ネットワークトポロジーを変える実験がカーネル, ユーザプログラム共に変更を加えることなく容易に行えた.

これらはカーネルの設計が妥当であることを示すものと考えられる.

7 おわりに

Transputerネットワーク上に記号処理言語用分散カーネルを作成し, 位置透過な非同期式通信と汎用的なごみ集めを実現した. 今後は通信量や負荷の統計をとる機能をカーネルに組み込み, 言語処理系やその他のプログラムの性質を評価することを考えている.

参考文献

[1] 前川, 所, 清水編: 分散オペレーティングシステム - UNIXの次にくるもの, 共立出版 (1991).

[2] *Transputer Reference Manual*, Prentice Hall (1988).

[3] Ichiyoshi, N., Miyazaki, T. and Taki, K.: A Distributed Implementation of Flat GHC on the Multi-PSI, *Proceedings of the Fourth International Conference on Logic Programming*, pp. 257-275 (1987).

[4] Nakajima, K., Inamura, Y., Ichiyoshi, N., Kurosawa, K. and Chikayama, T.: Distributed Implementation of KL1 on the Multi-PSI/V2, *Proceedings of the Sixth International Conference on Logic Programming*, pp. 436-451 (1989).

[5] Morris, F. L.: A Time- and Space- Efficient Garbage Compaction Algorithm, *Comm. ACM*, Vol. 21, No. 8, pp. 662-665 (1978).

[6] Nilsson, M. and Tanaka, H.: Fleng Prolog - The Language which turns Supercomputers into Parallel Prolog Machines, *Logic Programming '86 (LNCS264)*, Springer-Verlag, pp. 170-179 (1989).

[7] 横手: オブジェクト指向分散オペレーティングシステム, コンピュータソフトウェア, Vol. 9, No. 3, pp. 15-35 (1992).