6U-9

並列オブジェクト指向言語DinnerBell
——並列実行処理方式の評価——

立川江介　河野真治　田中英彦
東京大学　工学部

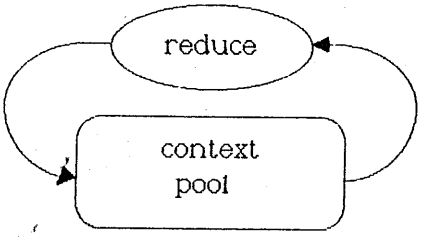
1.はじめに

ORAGAシステムはプログラミング言語として並列オブジェクト指向言語DinnerBellを使用している。DinnerBellではデータフローの考えをオブジェクト指向に取り入れることにより，プログラムに内在する並列性を自然な形で引き出して実行することができる。DinnerBellではソースプログラムはメソッドごとにコンパイルされコンテクスト〔図1〕と呼ばれる実行単位の列に展開される。

Destination	Message	Reply	ReplySelf

〔図1〕コンテクストの構成

コンテクストは各プロセッサごとにスケジューラのコンテクストプールに入れられ，実行はコンテクストをコンテクストプールから取り出し，対応するメソッドの示すコンテクスト列を再びスケジューラのコンテクストプールに加える事により進んでいく〔図2〕。DinnerBellでは並列実行における協調機構として単一代入則を採用していて，コンテクストの発火順序はこれらの単一代入変数を通してコンテクスト間のデータ依存性としてあらわに記述する。



〔図2〕コンテクストモデルを用いたDinnerBellの実行

2.コンテクスト実行に伴う処理

DinnerBellを実行する際において並列性は新たに生成される多数のコンテクストを複数のProcessor Element (PE) に分配し並列に処理することにより得られる。コンテクストは Destinationで示されるオブジェクトにMessage を送る動作に対応している。コンテクストの実行に伴う処理としては，

(1) Destination 及びMessage から対応するメソッドを探す。

(2) 引数の受け渡し，及び結果の戻り先をbindする。

(3) メソッド変数領域の確保及び新しいコンテクストの生成。

の3点よりなる。このうち(3)の処理は(1)，(2)の処理に比べて大きな手間がかかるため，(1)，(2)の処理だけで済むプリミティブメソッドは多数の新しいコンテクストを生成する一般の処理に比べかなり速い操作となりうる。

すでに作成した高並列性が得られることを確認する目的の簡易版シミュレータ[1]では

①PE間のネットワークにおける通信速度はプロセッサが一つのコンテクストを実行する速度に比べて十分に速いとする

②コンテクストはその内容の差異による処理時間の違いは考慮せずすべて一律に1サイクルで実行されるものとする。

という2つの仮定をもっていた。

(3)の処理を伴わないプリミティブメソッドなどでは(3)の処理を伴う一般のコンテクスト処理に比べ高速に実行されうるが，この場合新たなコンテクストの生成がないため処理の高速化を行っても①の仮定は破られない。

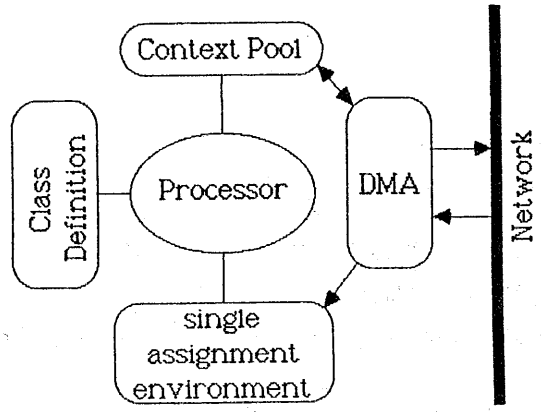
3.シミュレータ

各PEの構成としては図3のものを想定する。

コンテクストは多数のPE間で頻繁にやりとりされるデータとしてはオブジェクトなどの他のデータに比べ大きなものであるため，PE間の通信量の上からも一つのコンテクストを処理する間に新たに生成されるたすのコンテクストをすべて他のPEに送出することはその数が極めて多い場合はできない。一方前記のようにDinnerBellではコンテクストの分散により並列性を得ているため，コンテクスト分散戦略としては各PE間の通信量を上げずに高い並列性が得られる分散戦略が望まれる。

Concurrent Object Oriented Language 'DinnerBell'　— Evaluation of concurrent implemetation —

Kousuke TATSUKAWA, Shinji KOHNO, Hidehiko TANAKA
University of Tokyo



〔図3〕Processor Element (PE) の構成

このため簡易版シミュレータでは上記の仮定のもとで実行に必要なサイクル数及びネットワークを通してやりとりされるデータの量の測定を行った。

分散戦略

(1)単純分散

生成されたコンテクストをすべて各PEに順番に割り当てていく方式であり，他の方式との比較の際の基準とする。

(2)参照先分散

コンテクスト外に参照がある場合，その参照しているPEにコンテクストを送る方式であり，外部参照の回数を減らし各PE間のデータの転送量を減らすことを目指している。

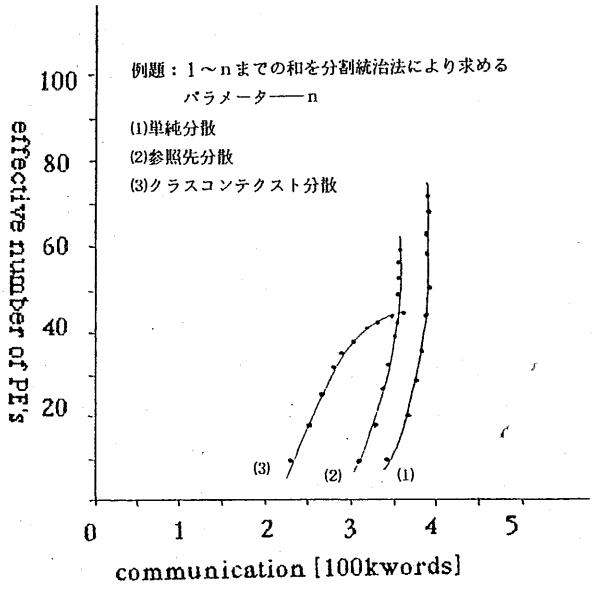
(3)クラスコンテクスト分散

DinnerBellの場合クラスを Destinationにもつコンテクストは多数の新しいコンテクストを生成することが多いため，そのようなコンテクストだけを他のPEに送出する方式である。これにより各PEから外部へ送出されるコンテクストの数を抑えることができ，通信量を減らすことができる。

4.評価

簡易版シミュレータによる測定結果を図4に示す。コンテクストはデータに比べ通信単位としては大きいため参照先分散よりクラスコンテクスト分散の方がより通信量の軽減が得られている。

また参照先分散ではオブジェクトを持つ特定のPEにコンテクストが集中するため通信量が減らないわりに実行に要するサイクル数の増加が大きいが，クラスコンテクスト分散では生成されるオブジェクトも各PEに分散させることができるためその点でも優れている。



〔図4〕通信量と並列度の関係

5.現状

前記のシミュレータではかなり大きな仮定をおいているためより詳細なシミュレーションを行う必要がある。特にプリミティブメソッドの実行を速くした場合の影響や，ネットワークの影響などは調べる必要がある。

新たに作成したシミュレータでは簡易版のシミュレータに比べ

(1)コンテクストの実行に要する時間がコンテクストにより異なり各PEが非同期に動作する。

(2)ネットワークの影響を考える。

・通信に要する時間

・ネットワークの遅延

の2点について考慮したシミュレーションを行うことができる。

現在はこのシミュレータを用い，コンテクストモデルをもちいた並列オブジェクト指向言語DinnerBellの実行のより詳細なシミュレーションを行い，測定結果の評価を行っている。

6.参考文献

〔1〕情報処理学会第33回全国大会，5D-3 (1986)