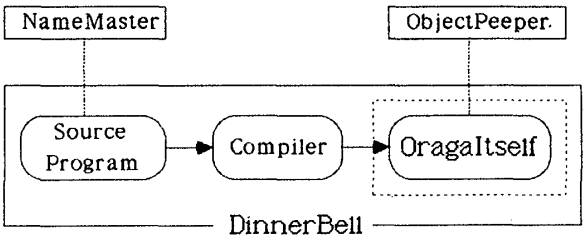
5D-3

並列オブジェクト指向システムORAGA
— 並列実行のシミュレーション —

立川江介　渡部眞幸　河野真治　田中英彦
東京大学　工学部

1.はじめに

ORAGAシステムでは，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 を用いる。 DinnerBell の処理系としては，入力されたソースプログラムを Compiler によりコンテクストという実行単位の列にコンパイルし，それをアーキテクチャにより直接に実行する形態を考える（図1）。コンテクストは，メッセージを引数単位に分解したものに対応している。



（図1） DinnerBell の処理系

言語 DinnerBell については，すでに擬似並列的に動くインタプリタが作成され動作している〔1〕。我々はORAGAシステムを支えるアーキテクチャである OragaItselfを試作するための準備として，まず DinnerBell の動作状態をより詳しく知るためにソフトウェアシミュレータを作成した。以下では，ソフトウェアシミュレータの構成及び，並列実行のシミュレーションについて述べる。

2.DinnerBellの実行

DinnerBell の処理系ではコンテクストを基本単位として実行が進んでいく。コンテクスト（図2）は，Destination,Reply,ReplySelf 及び Selector と Argument により構成される一引数のメッセージよりなる。

Destination	Message		Reply	ReplySelf
	Selector	Argument		

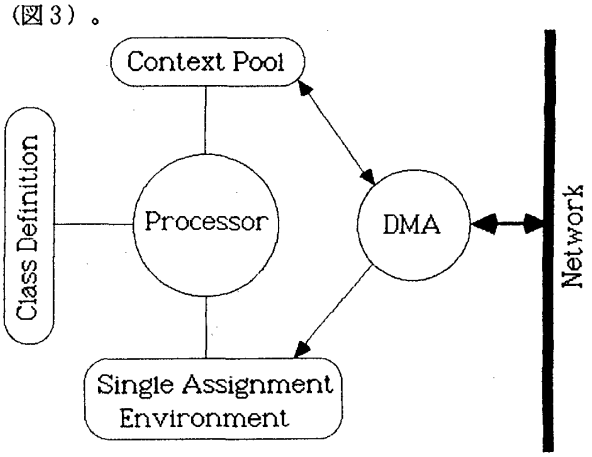
（図2）コンテクストの構成

コンテクストの実行とは，ある一つのコンテクストから，あらかじめコンパイルされたクラス定義に基づいて新たに複数個のコンテクストを生成することである。ORAGAシステムでは，多数のコンテクストを複数のPE（Processor Element）に割り振って処理を行うことによりプログラムを並列に実行する。メッセージを引数毎に分解してコンテクストとしたのは，単一のメッセージでも各引数毎に独立に実行を行うことにより引数間の並列性を引き出すためである。

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る。コンテクストの各要素はこの単一代入変数を指し示していて，コンテクスト間のデータ依存性はこれらの単一代入変数を用いてあらわに記述される。

3.ソフトウェアシミュレータ

ここではソフトウェアシミュレータの想定している動作環境について若干の説明を行う。 OragaItselfのPEは，クラス定義，オブジェクトの環境，未処理のコンテクストを格納しておくコンテクストプールのメモリ領域と，コンテクストの変換を行うためのプロセッサとからなる（図3）。



（図3） Processor Element

Concurrent Object Oriented System 'ORAGA' — Simulation of concurrent execution

Kosuke TATSUKAWA, Masaki WATANABE, Shinji KOHNO, Hidehiko TANAKA
The University of Tokyo

各PEはネットワークで結合され，PE間に共有メモリはない。PE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十分に速いため，ネットワーク自信がネックになることはないと仮定する。

シミュレータはミニコンピュータ VAX11／730 上に実装されていて，C言語で約2000行ある。シミュレータの構造上，コンテクストの内容の差異による処理時間の違いは考慮せず，すべてのコンテクストは一律に1サイクルで実行されるものと仮定した。また，PE間の通信は簡単のためどのPE間でも一定の遅延でデータ転送ができるものと仮定する。

4.負荷分散の方針

OragaItselfでは，コンテクストを各PEに分配することによりプログラムを並列実行しているため，PEの実行効率はコンテクストを各PEに分散する戦略に依存する。この戦略を決定するためのポイントとしては

①各々のPEにおいて，処理されるコンテクストの数
（PE の負荷）
②PE間でネットワークを通してやりとりされるデータ量
（ネットワークの負荷）

の二つがある。ネットワーク上で通信されるデータとしてはコンテクストの通信（分散），Reply,Replyself の返信，及びそれらに伴うコントロール信号がある。

各PEにコンテクストが平均して分散し，さらにPEの外部への参照をなるべく生じないようなコンテクストの分散戦略が望ましい。以下に示す三つの分散戦略について検討を行っている。

（1）単純分散

これは，データ転送については考慮せず，生成された全てのコンテクストを各PEに順番に割り当てて行く方式である。この方式は他の方式との比較の際に基準としてもちいる。

（2）ルーチン分散

一つのルーチンとしてまとまった内容を持つコンテクストをまとめて一つのPEに送る方法である。この方法では一つのルーチンの中の処理は一つのPEの中でまとまって行われ，計算結果などの一部のデータだけがネットワークを介して戻ってくるため，処理を行っているPEから外部への参照の回数は大幅に少なくなることが予想される。この方法を用いた場合，分割統治法を用いて問題を解く際に，分割した問題をそれぞれ一つずつのPEにまとめて割り当てることにより，効率的に実行を行うことができると考えられる。

（3）参照先分散

コンテクスト内に外部への参照がある場合に，その参照先のPEにコンテクストを送ることにより，外部参照の回数を減らす方法が参照先分散である。しかし，この方法は，初期状態ではコンテクストが一つのPE内だけに存在し，外部への参照が存在しないため，起動時からこの方法を用いることはできない。このため，参照先分散を用いる場合には，あらかじめ別な方法によりコンテクストを各PEにある程度分散させとおいてから切り換えて使う必要がある。

5.シミュレーション

現在，4.で挙げたコンテクストの各分散戦略について，簡単な例題について並列実行のシミュレーションを行い，シミュレーション結果を評価中である。今後はこのソフトウェアシミュレータの結果に基づいて，ハードウェアシミュレータの細部の仕様を決定し，その作成を行う予定である。

6.参考文献

〔1〕情報処理学会代32回全国大会，6F-1（1986）