分散処理のためのC++言語拡張とその実現方法

6J－1

浅野郁久　安岡義弘　上田賀一

茨城大学工学部情報工学科

1　はじめに

ネットワーク上に存在するさまざまな資源を有効利用するために、分散処理を導入することが考えられる。しかし、分散処理を意識したプログラムを記述するためには、ネットワークやプログラミング言語上での実現方法などについての詳細な知識が必要となり、記述する量も増大する。

本研究では、この様な分散処理をC++を用いて行おうとする際に増加する負担の軽減する方法について述べる。

2　分散方法

分散プログラムを記述する際のプログラマにかかる負担を軽減させるための様々な研究が行われている。

DOL/C++[1]では、「通信者」と「伝達者」というクラスを用いて分散したプロセス間の通信の問題を解決している。

オブジェクト管理サーバ[2]はC++でのオブジェクト(インスタンス)を分散の単位としプロセス化し、サーバを用いてこれらを管理している。これらは、分散処理用のライブラリを用意することで、記述量を減らすことを行っている。しかし、既存のC++プログラムを分散させるためには多くの変更を要する。

また、WARASA[3]では、自律/排除/条件同期オブジェクトというオブジェクト型を用いて、基本的な並列機能を分担することで並列機能記述の隠蔽を行っているが、これはMach OSのプリミティブを利用しているため並列機能のみで多機種分散はできない。

そこで本研究では、多機種分散が可能なオブジェクト管理サーバの考えを基に、クラス管理サーバを設計し、また記述量を減らすために拡張C++を設計する。

Extension of C++ language for distributed processing
Fumihisa Asabno, Yoshihiro Yasuoka, Yoshikazu Ueda
Department of Computer and Information Sciences, Faculty of Engineering, Ibaraki University

3　クラス管理サーバ

3.1　クラス管理サーバ方式

プログラム中で生成されるインスタンスをクラス単位で分散させ、各クラスを一つのプロセス(プロセス化クラス)とし、その中にインスタンスを生成する。

クラス単位で分散することにより、インスタンス単位でプロセスを作る場合より、プロセス数の増加に伴うサーバへの負担が抑えられる。また、メソッドをメモリ上の一箇所に置くことができる。

反面、クラスを単位とするために個々ののインスタンスの独立性が失われることになる。また、ある特定のクラスのインスタンスのみが多量に生成された場合、分散の割合が下がってしまうことになる。

3.2　構成

クラス管理サーバ方式は、プロセス化クラス、クラス管理サーバおよびスーパバイザサーバ(SVS)から成る。概観図を図１に示す。

プロセス化クラスの管理はクラス管理サーバによって行われる。クラス管理サーバの機能は、クラスの生成,消去、プロセス化クラスとの通信を行うことである。

またクラス管理サーバは、SVSにより管理される。SVSの機能としては各クラス管理サーバの起動、終了や各プログラムでのグローバルな変数の値の管理などを行うことである。

4　拡張C++

C++言語を基本言語とし、コメント領域に所定の文字列を記述することでクラスのインスタンスの分散を指示する拡張を行うことにした。拡張部分を基本言語のコメントとして記述することで基本言語C++の構文の変更が行われず、通常のC++コンパイラでもコンパイルできるようになる。

書式

```
//:DISTRIBUTE < TYPE >
instanceptr = new class_name
```

または

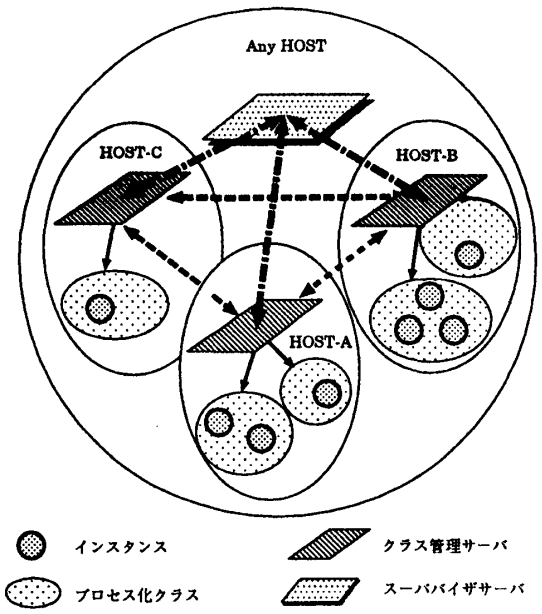

図 1: クラス管理サーバ方式の概観図

```
//:DISTRIBUTE < TYPE >
class_name instace_name;
```

インスタンスを生成する new 文または宣言の前の行に記述する。TYPE にはどのように分散するかを指定する。

拡張 C++のコンパイラはフィルタの形式をとり、拡張C++のプログラムソースファイルをフィルタを通してプログラムを C++形式に変換し、C++コンパイラに渡すことで実行形式のファイルが作られる。

制限としてはメソッドおよび関数呼び出しでは引数の変数引数渡し (ポインタ) を禁止する (複数のマシン間でアドレス空間が違うため)。ただし、文字列を示すための char 型のポインタは使用できる。また、変数および定数の宣言はブロックの先頭で行うものとする。

5 フィルタ

フィルタで行われる処理は、以下の通りである。

- クラス宣言毎のファイルの分割。
- 各ファイルへの通信機能の付加。
- クラスのインスタンス生成、削除のライブラリを用いた形式への変換。
- メソッド呼び出しとメソッドからの返り値の変数への代入のライブラリを用いた形式への変換。

ファイルの分割は 1 つのクラスを単位として行い、それぞれのクラスがプロセスとして動作可能な様に処理をファイルに加える。また、クラス管理サーバとの通信のための機構と、クラスの各メソッドに対して共有メモリにアクセスするための機構も付加する。

クラスのインスタンス生成に対する処理は、そのインスタンスが分散の対象となっているかどうかを判断し、対象となっているならばその指定にしたがって置き換えを行う。削除はライブラリにより用意される関数を使って行う。

メソッド呼び出しの処理は、その引数の型についての情報をもとに呼び出すメソッドを推定し、ライブラリにより用意される関数を用いて行うこととする。引数の型についての情報を得るために、変数の宣言についての処理が必要である。

既存の C++コンパイラを利用することでシステムを小さくすることができるので、拡張 C++から C++に変化するためのフィルタを用意する。また、プリプロセッサを通したものをフィルタの入力とすることで、更にフィルタの負担を減らせる。また、独自のプリプロセッサ命令を用いている場合でも処理の対象とすることができる。なお、フィルタでは C++でのアクセスレベルなどの規則についてはチェックを行わず、C++コンパイラに任せることにしている。

フィルタの実装は拡張や変更の容易さから flex および bison を用いて行っている。

6 おわりに

C++で分散処理を行うための方法として、クラス管理サーバ方式とこれを用いるためのフィルタについて述べた。今後は、ポインタの扱いについての制限の解消などについて検討を加えたい。

参考文献

[1] 鈴木 寿郎, 中澤 修 : “分散ソフトウェア構築のための DOL/C++クラスライブラリ”, 情報処理学会研究報告, vol.94, No.56, pp.25–30 (1994.7)

[2] 新井 俊行 : “オブジェクト指向概念に基づいた分散処理に関する研究”, 平成 5 年度茨城大学大学院工学研究科修士論文 (1994.2)

[3] 江 允, 牧之内 顕文: “WARASA:軽量プロセス上での並列オブジェクト指向プログラミング言語, 情報処理学会論文誌, vol.35, No.7, pp.1342–1351 (1994.7)