並列記述の隠蔽可能なオブジェクト指向言語*

6L－5

鈴木 孝一郎 阿刀田 央一 冨澤 眞樹 田村 仁
(東京農工大学　工学研究科　電子情報工学専攻)

1　はじめに

現在、多くの並列オブジェクト指向言語が存在する。しかし、その多くは各オブジェクトが並列に動作するオブジェクトモデルを採用しているため、クラスの利用者が、オブジェクトの並列動作をコントロールするコードを記述しなくてはいけない。

クラスライブラリ自身に並列コードを埋め込み、そのオブジェクト自身による自発的な並列処理を行なうことができれば、クラスの利用者は、並列コードを記述することなく並列計算機のパワーを引き出せるだろう。

そこで、並列記述をクラスの中に封じ込め、クラスの利用者から並列記述を隠蔽するためにオブジェクト指向言語を利用する。

2　並列記述の隠蔽

オブジェクト指向言語は、クラスの利用者と作成者の境界が厳密な言語である。そこで、C++[1] のクラス定義に並列記述を隠蔽する仕組みを提供する。この並列記述の隠蔽により、クラスの利用者は、並列を記述せずに並列処理を行なえる。また、クラスの作成者も、並列性がクラスの外部に現われないため、並列クラスを組み合わせた複雑なクラスを作ることができる。

そこで、C++のクラス定義に並列記述を許し、並列度の高いクラスライブラリを構築するための言語 Cs++ を設計した。

3　設計方針

Cs++を設計するにあたって、次のような方針をたてた。

1. 共有メモリ方式の並列計算機をターゲットとする
2. 抽象的なインタフェースをもつこと
3. ベースとなる言語への変更を最少にすること

*An object oriented language which encapsulates parallel codes,
Koichiro Suzuki ,Oichi Atoda ,Masaki Tomisawa,Hitoshi Tamura
Tokyo University of Agriculture and Technology,
2-24-16,Nakamachi,Koganei,Tokyo

4　C++の拡張

Cs++は、構文規則の追加をなくすため、すべての並列制御にかかわる操作はクラスライブラリを介して行う。追加されるクラスは、Transfer クラス、Among クラス、System クラスの3つである。さらに、共有変数の指示や排他制御も、構文規則の変更はせずに、意味規則の解釈の変更でおこなう。

4.1　クラスの追加

1. Transfer クラスは、非同期にメンバ関数を実行するために、並列分流と合流をするメンバ関数を持つ。また、そのオブジェクトは、同時にインスタンスの識別子でもあり、インスタンスの状態を表すステータスでもある。
以下にトランスファクラスの並列分流と合流のメンバを示す。

```
// 返り値が RETTYPE, 引数が一つで ARG1TYPE で
// あるクラス CLASS のメンバ関数を並列呼び
// するためのトランスファクラスのテンプレート
template <class CLASS,class RETTYPE
                    ,class ARG1TYPE1>
class Transfer {
public:
// メンバ member を並列呼び出しするメンバ関数
 int askfor(
   RETTYPE (CLASS::* member)(ARG1TYPE));
//並列呼び出しをした関数と同期、返り値を得る
   RETTYPE thank(void);
}
```

2. 複数の並列呼出しの終了順に処理を続けたいことはよくあることである。Among クラスは、複数のトランスファオブジェクトを登録し、終了したものを返す。

```
Transfer<Foo,int,int> t[10];
Among amg;
...                    // 並列呼び出しを行う
```

```
// amg にトランスファオブジェクトを登録
for(int i=0;i<10;i++)

    amg.regist(t[i]);
    // 始めに終った呼び出しを待つ
    i = amg.wait();
    // t[i] から返り値を得る
    ans=t[i].thank();
```

3. Cs++では、接続されている物理プロセッサ数を知らずに並列呼出しを行える。しかし、並列処理の最適化をするためには、プロセッサ数を知る事が必要な場合がある。そこで、プロセッサの数を知るための System クラスを用意した。

```
System system ;
// プロセッサの数を返す
penum = system.penum();
```

4.2 意味規則の変更

変更された意味規則は、1. トランスファクラスの位置の制限、2. 暗黙的な排他の規則、3. スコープの解釈についてである。

1. Transfer クラスの位置の制限

   Cs++は、並列クラスを構築する事が目的なので、関数の内部での並列分流や合流は望ましくない。そこで、Transfer クラスの使用はクラス定義の内部でだけ許される。

2. 暗黙的な排他の規則

   オブジェクトのデータメンバは、メンバ関数が並列に動作するため、不正に変更される危険性がある。ところで、C++で const メンバ関数とは「オブジェクトの状態を変更しない」関数という意味がある。Cs++では、その意味を変更、オブジェクトの状態を変更しない const メンバ関数は並列で実行されるが、オブジェクトの変更をする可能性がある const でないメンバ関数は、逐次に実行される。

3. スコープ規則の解釈

   オブジェクトのメンバ関数は並列に実行されるため、そのデータメンバは複数のプロセッサから参照・変更が行われることになる。そのため、全てのオブジェクトのデータメンバは全てのプロセッサに共有される。また、同様に、グローバルなデータも共有される。

5 記述例

簡単な例として、行列のかけ算をクラスを示す。

```
class Matrix {
//並列で実行されるメンバ関数
void multi_sub(Matrix&b,Matrix&ret,int row){
  for(int i=0;i<N;i++)
    for(int j=0;j<N;j++)
     ret[row][i] += elem[row][j]*b.elem[j][row];
  }
public:
  int elem[N][N];
//利用者から呼ばれるメンバ関数
Matrix multiple(Matrix & b){
//返り値が void,Matrix&,Matrix&,int が引数の Transfer
  Transfer<Matrix,void,Matrix&,Matrix&,int> t[N];
  Matrix matrix;
//列ごとに並列呼びを行ない、演算
  for(int i=0;i<N;i++)
   t[i].askfor(multi_sub,b,matrix,i);
//制御フローの合流
  for(i=0;i<N;i++)
   t[i].thank();
  return matrix
  }
};
```

6 言語の実装

Cs++を、ネットワークで接続された複数のワークステーション上で実装した。リモートマシン上で非同期にメンバ関数を実行する機構は、sun の RPC と fork の組合せで実現した。また、クラスのデータメンバや外部データ等の共有データは、共有データを保持するサーバを作り、一元管理する。サーバに保持される情報は、アーキテクチャに依存しない形で保持されるため、異なる計算機間で共有変数を実現できる。

7 まとめ

本稿では、クラス定義に並列記述を記述し、クラス内部で並列化を行なう言語 Cs++を設計について述べた。この言語により、並列記述が内部で完結しているクラスを記述することができる。さらに、それらのクラスを組み合わせて新しいクラスを記述すれば、より高速なクラスライブラリを記述することができる。

参考文献

[1] Margaret A.Ellis and Bjarne Stroustrup. *The Annoteted Reference Manual.* Addison-Wesley, 1992.