ＱＣＤＰＡＸのプリプロセッサとベクトルコンパイラ

3T-8

駒井　寛*1　澤崎　武*2　星野　力*2
(*1アンリツ㈱　*2筑波大学)

1．はじめに

現在作製中の高並列計算機ＱＣＤＰＡＸは、浮動小数点演算を高速に行う機能を備えさらにベクトル演算が実行可能である。ＰＵ(Processing Unit)のアレイ構造とベクトル演算をサポートする高級言語ｐｓｃ[1]の改良及び、ホスト計算機(Sun3/260)とＰＵを結ぶためのホストプログラム用プリプロセッサと関数ライブラリの作成を行った。本報告ではｐｓｃコンパイラとホストプログラムについて述べる。

2．ｐｓｃ

ｐｓｃ(Parallel Scientific C)は科学技術計算を記述するための言語である。また、ｐｓｃはアレイ型計算機の各ノード(PU)記述の言語である。制御文や式の優先順位などはｃ言語と同等である。ポインタは関数の引数にのみ使用でき、構造体、共用体、文字型などはサポートされていない。

2.1. 変数の領域と型

ｐｓｃの変数はfast,slow,east,west,south,north,ferryのいずれかの領域に属する。fastはSRAM、slowはDRAM、east,west,south,northは東西南北の各ＰＵとのＣＭ(Communication Memory)、ferryはホスト計算機との２ポートメモリに割り当てられる。また変数の型として整数型(int)、実数型(float)、複素数型(complex)の３つの型があるが複素数型は現在まだサポートしていない。変数宣言では領域名と型を指定する。ホストとのデータ通信には大域変数を用いる。ferry領域の大域変数はＰＵが動作中でもホストからアクセスできるが、他の領域の大域変数はＰＵがホルト状態の時のみアクセス可能である。

2.2.初等関数とプリミティブ関数

ｐｓｃは、数値計算用の初等関数と、並列処理を実行する際に必要な機能をプリミティブ関数としてもつ。プリミティブ関数を表１に示す。これらのプリミティブ関数はコンパイラがインライン展開して出力する。

表１．プリミティブ関数

pusync 全PUの同期	pubreak 全PUのホルト
puxnum PU番号のx方向	puynum PU番号のy方向
latchtm　内蔵タイマーラッチ	
resettm　内蔵タイマーリセット	

2.3.ベクトル演算

ここでのベクトル演算とは、ある一連の演算を実行時に制御メモリ(WCS)に書き込むことにより、浮動小数点プロセッサ(FPU)がＦＰＵコントローラの制御によってCPUから独立して行う演算をいう。したがってベクトル演算中は、CPUでfast領域を使わない別な演算を並列に実行可能である。ｐｓｃではこれをvfor-do文により実現する。vfor-do文の例を図１に示す。vfor以下のブロックがFPUのための記述でありdo以下のブロックがCPUのための記述である。先に演算を終了した方が他方の演算終了を待ち、両方が終わった時点で次の演算を実行する。

```
vfor( v = vstart  ; v < vmax ; v += vstep )
  sp[v*2+s] = m[v] * la[v][p] + sp[v*2+s]
                    * ( one - la[v][p] ) ;
do {
  for( j = 0 ; j < 10 ; j += 1 ) {
    l_east[j] = l[j] ;
    k[j] = k_west[j] ;
  }
}
```

図１．vfor-do文の例

3．ｐｓｃコンパイラ

ｐｓｃコンパイラは１パスであり、浮動小数点演算部分はｑｃｄｐａｘ専用のアセンブリ言語ｑｆａを、その他の部分はMC68020のアセンブリ言語を出力する。

3.1.ｐｓｃプリプロセッサ

現在はまだ複素数型をサポートしていないため、複素数型変数を実数型の配列に置換し、複素数型演算を実数型の演算に展開するプリプロセッサを作成した。

4．ホストプログラム

ＱＣＤＰＡＸでは、ｐｓｃで書かれたプログラムをホスト計算機上でコンパイルとアセンブルを行った後、ホストプログラムによってＰＵにロードし実行する。

Vectorizing Compiler and its Preprocessor for QCDPAX
Hiromu KOMAI[1], Takeshi SAWAZAKI[2], Tsutomu HOSHINO[2]
[1]Anritsu Corp., [2]Univ. of Tsukuba

ホストプログラムはｃ言語で記述し、主として、ＰＵとの入出力インターフェイスと、ＰＵとグラフィックディスプレイの制御の２つの役割をもつ。ホストプログラムとＰＵ用プログラムの例を図２に示す。

4.1.ホストプログラム用プリプロセッサ

このプリプロセッサは、ＰＵとホスト間のデータの入出力を容易に記述するためのものである。プリプロセッサを用いることにより、ＰＵ用プログラムで宣言された大域変数を、ホストプログラムでも同様に宣言すれば、双方から同一の変数名でアクセスが可能になる。また、ｃ言語でサポートしていない複素数型変数を、実数型のメンバをもつ構造体に置換する。さらに、複素変数の四則演算を、関数に書換え実数型に対する演算とする。

4.2.ホストプログラム関数ライブラリ

関数ライブラリは、ＰＵ制御用とグラフィックディスプレイ制御用の２つがある。ＰＵ制御関数の主なものを表２に示す。

5．おわりに

ＰＵ用の高級言語ｐｓｃのコンパイラの改良と、ホスト用プリプロセッサ、関数ライブラリの作成を行った。現在、複素数演算をサポートするようコンパイラを拡張中である。現在のコンパイラはベクトル演算部の出力が冗長であるため、ユーザが手で最適化を行っている。今後、コンパイラによる最適化が必要である。

本研究は科学研究費特別推進研究(62060001)による。

[参考文献]

[１]藤井、斉藤、星野：ＱＣＤＰＡＸのコンパイラ
情報処理学会第３６回全国大会　７Ｃ－６

表２．PU制御関数

connect	PUとホストを接続
sellectpu	PUの使用範囲指定
resetpu	PUのリセット
loadprog	PU用プログラムをPUにロード
startpu	PUの実行開始
haltpu	PUをホルト
relhaltpu	PUのホルト解除
waitpu	PUの処理を待つ
getpustate	PUの状態を得る

```
#include  <qcd_host.h>
fast  complex a[5], sum, pusum, tmp;
north complex sum_n;
east  complex sum_e;
south complex sum_s;
west  complex sum_w;

main() {
int j, k, l;

  connect();
  selectpu( ALL_HLB, ALL_MLB, ALL_LLB );
  resetpu();
  loadprog( "csum.p" );
  sum = cmplx( 0.0, 0.0 );
  for( j = 0, j < 4, j++ ) {
    selectpu( ALL_HLA, ALL_MLA, j );
    for( k = 0, k < 4, k++ )
      a[k] = cmplx( (float)j, (float)k );
  }
  selectpu( ALL_HLB, ALL_MLB, ALL_LLB );
  startpu();
  waitpu();
  for( j = 0, j < 4, j++ ) {
    selectpu( ALL_HLA, ALL_MLA, j ) ;
    printf( "PU=%d pusum=(%-6.1f,%-6.1f)
          sum=(%-6.1f,%-6.1f)¥n", j, pusum, sum );
    printf( "north=(%f,%f) east=(%f,%f) south=(%f,%f)
          west=(%f,%f)¥n", sum_n, sum_e, sum_s, sum_w );
  }
}
```

← 同一の大域変数宣言 →

```
fast  complex a[5], sum, pusum, tmp;
north complex sum_n;
east  complex sum_e;
south complex sum_s;
west  complex sum_w;

main() {
fast int  i;

/* get the sum in the PU */
  sum = a[0];
  for( i = 1, i < 5, i += 1 )
    sum = sum + a[i];
  pusum = sum;

/* get the sum over all */
  pusync(10);
  sum_w = sum;
  pusync(15);
  tmp = sum_e;
  sum = sum + tmp;
  pusync(20);
  sum_s = sum;
  pusync(25);
  tmp = sum_n;
  sum = sum + tmp;
}
```

ホストプログラム　　ＰＵ用プログラム

図２．ホストプログラムとＰＵ用プログラムの例