MUV GRAPE-7 移行メモ

斎藤貴之 (国立天文台天文シミュレーションプロジェクト 研究員)

2007 年 6 月 18 日

概要

このメモは、国立天文台 GRAPE システム 通称 MUV (Mitaka Underground Vineyard) で GRAPE-5 を使っていたユーザが、国立天文台が新たに導入した GRAPE-7 からなるシステムに速やかに移行するための手続きをまとめたものである。

1 はじめに

国立天文台 GRAPE システム、通称 MUV (Mitaka Underground Vineyard) は 2001 年 1 月、天文学データ解析計算機センター (当時) より共同利用計算システムとして、日本の天文コミュニティーに公開されてきた。現在は、2006 年 4 月に発足した天文シミュレーションプロジェクトによって運営されている。MUV では無衝突系を扱う GRAPE-5 と衝突系を扱う GRAPE-6 の二系統が運用されている。

しかしながら、GRAPE-5/GRAPE-6 ともに開発より 10 年近く経て性能の陳腐化が進んでいる。特に、近年の性能飛躍 (アーキテクチャの更新とマルチコア化) により、ものによっては ∼ 20 Gflops という性能を持つ汎用 CPU が存在するに至る。例えば Phantom-GRAPE [*1] は、アセンブラレベルのチューニングによってほぼカタログどおりの性能を出すことができる。コモディティーである汎用 CPU は、今後も性能躍如と低価格を維持すると思われる。一方 GRAPE-5 の性能は ∼ 40 Glops である。この値は最新の (しかしそれなりに廉価な) 汎用 CPU にたいしてもはや二倍程度高速であるに過ぎない。

国立天文台では、最先端の研究及び計算を実行できる環境を維持するため、定期的に MUV の更新を行っている。前回の更新 (2004 年 5 月) ではホスト計算機の更新を行った。今回の更新では、MUV の無衝突系部分の更新を行った。この更新ではホスト計算機の更新のみに留まらず、GRAPE-5 を最新の無衝突系専用 GRAPE である GRAPE-7 へと更新した。

このメモでは、従来の MUV GRAPE-5 系を利用していたユーザが速やかに GRAPE-7 に移行できるようにと用意したものである。適宜参考にして、最新の計算環境を手に入れてほしい。

2 GRAPE-7

GRAPE-7 は、GRAPE-5 の後継機として東京大学 福重俊幸助手 (当時) によって開発され、現在 K&F Computing Research 社[*2] により、開発・販売が行われている最新の GRAPE である。GRAPE-5 の後継機ということで無衝突系重力多体問題に特化している。

カスタムチップとして実装されてきた従来の GRAPE と異なり、GRAPE-7 はアルテラ社製 FPGA(Flexible Programable Gate Array)、CycloneII 上に GRAPE-5 のパイプラインを構築したものである。現在のところ、model 100, model 300, model 600 (それぞれ、∼ 101 Gflops,∼ 228 Gflops,∼ 456 Gflops) の三種類が存在する。今回 MUV に導入されたのは、このうち最も高性能な model 600 である。

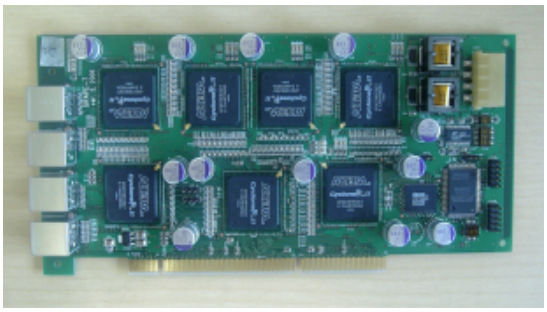

図 1 GRAPE-7 model 600。写真は K&R Computing Research の web page より。一部改変。

[*1] URL:http://grape.astron.s.u-tokyo.ac.jp/~nitadori/phantom 重力多体系計算ルーチンのクリティカルパスをアセンブラレベルでチューニングし潰している。Intel Core 2 Duo 2.4GHz において、18.3 Glops を記録している (2007 年 6 月現在)。

[*2] URL:http://www.kfcr.jp

3 MUV GRAPE-7 利用のための環境について

3.1 システム

GRAPE-7 のシステムは、従来の MUV GRAPE-5 系と似た構造を持つ。つまり、LSF サーバとして `grapex.cc.nao.ac.jp` (Intel Xeon 3.0GHz, 2cores, 1GB memory) が、LSF ノードとして `bunch01x-bunch16x.cc.nao.ac.jp` (Intel Xeon 5160 3.0 GHz, 2cores, 4GB memory) がある。

コンパイルなどはすべて `grapex` 上で行う必要がある。今システムより OS が 64 bit になった。従来の `grape/bunch01-16` は、32 bit システムであったため、バイナリ互換がない。そのため、**必ず** `grapex` 上でコンパイルを行う必要がある。なお、`grapex` では、コンパイラとして `gcc 3.4.5`、`icc 9.1`、`ifort 9.1` が提供されている。

ジョブ投入方法は `grapex` 上で従来同様 `bsub` コマンドを使って投入する。**ただし、現在 LSF ライセンスは新システムに移行されていない。**移行後は必ず `bsub` を使ってもらうが、特にアナウンスがあるまで各 `bunch??x` にログイン後直接実行してもらいたい。**当面は MUV におけるいわゆるデバッグノード扱いとなる。**

3.2 ライブラリパス

GRAPE-7 のライブラリは次のようになっている。

```
/usr/local/g7pkg
```

したがって、インクルードファイルパス及びライブラリパスは

```
include path : /usr/local/g7pkg/include
library path : /usr/local/g7pkg/lib
```

となる。

OS のシステムと独立にインストールした `grapex/bunch01x-bunh16x` 用の gsl(GNU Science Library 1.8) と pgplot は次のパスにそれぞれインストールされている。

```
gsl : /usr/local/g7
pgplot : /usr/local/g7pgplot
```

4 GRAPE-7 への移行

以下、既に GRAPE-5 をつかって計算ができる状況にあると仮定して話を進める。なお、参考にしたライブラリのバージョンは GRAPE-5 用ライブラリが 1.7.2、GRAPE-7 用ライブラリが 1.3 [*3] である。

4.1 互換性

GRAPE-7 は GRAPE-5 互換として開発されたため、基本的に関数名は GRAPE-5 のものを踏襲している。ヘッダファイル名も、GRAPE-5 と同様 `gp5util.h` である[*4]。基本的な利用に関しては以下にあげるような点に注意すれば十分である。

まず、GRAPE-7 のパイプラインは GRAPE-5 と異なる。GRAPE-5 は実パイプライン 12 本、仮想パイプライン 96 本であり、シミュレーションコードによっては、96 本のパイプラインがあると決め打ちで書かれていると思われる。GRAPE-7 は model によって異なり 20 本から 120 本までの差がある。MUV に導入した GRAPE-7 は model 600 であったため、最も簡単にはパイプライン数を 120 本に変更する必要がある。賢明なプログラマであれば、パイプライン数取得関数 `int g5_get_number_of_pipelines(void);` を使って書いてあるかもしれない。この場合は問題にならない[*5]。

次に、GRAPE 上に保持できる j 粒子の数が異なる。GRAPE-5 では、最大 131071 個の粒子を一度に保持することができた (慣例に従いこの粒子数を `JMEMSIZE` と表記する)。GRAPE-7 における `JMEMSIZE` は回路規模、model、およびパイプライン最適化度合に依存する。よって、最大保持可能 j 粒子数取得関数 `int g5_get_jmemsize(void);` をつかう必要がある[*6]。もし、配列などでサイズが `JMEMSIZE` に寄るように書いてある場合は、動的確保するように書き換えておく必要がある。

GRAPE-7 では近傍粒子探査機能がなくなっている。利用者は各自ツリー法 [1, 2] などを用いて近傍粒子探査をホスト計算機で行う必要がある。

Particle Mesh (PM) 法で用いるための関数群は最新版の GRAPE-7 ライブラリ (version 2.0) から提供される。MUV のライブラリアップデートが行われ次第使えるようになるはずである。こちらの機能を使っているユーザはしばしお待ちいただきたい。

また、GRAPE-7 ではポテンシャルエネルギーを返さない。注意が必要である。

[*3] 2007 年 6 月 9 日現在、最新版は 2.0 である。ただしこちらはまだ MUV には導入されていない。

[*4] GRAPE-7 のマニュアル [5] によると、`gp5util.h` は `g5util.h` へのシンボリックリンクになっていることから `g5util.h` を使うことがお勧めされている。

[*5] 著者が調べたところ、パイプラインの数として 256 が返ってきた。現在の仕様は web 情報と異なるようである。

[*6] 著者が調べたところ、`JMEMSIZE` として 12288 が返ってきた。これは web 情報と一致するようである。

4.2 GRAPE-5/GRAPE-7 ライブラリ関数対応

表 1 に関数の対応表を用意した。紙面の都合上良く使われると著者が判断した関数に限った (割愛された関数の多くが著者ですら使ったことがない、存在を知らなかったものであったりする)。おもな関数を左に挙げ、その関数がそのまま使える場合は '○' を、つかえない場合は '×' を付けている。別な関数で代用可能なものについては、'×' を付けた後、脚注でどのようにすべきかを記載した。関数の機能の詳細については GRAPE-5/GRAPE-7 のマニュアルあるいは開発者の論文を参考にしてもらいたい [3, 4, 5]。

一瞥してわかるように、ほとんどの関数が同じ名前で使うことができる。他にこの表からわかることとして、近傍粒子探査機能に関する関数が用意されていないことと、Particle–Mesh 法用の関数が現在導入されているライブラリバージョン 1.3 では用意されていないことである。これらは全て §4.1 で述べた通りである。

表 2 には新たに追加された関数のうち主要なものを挙げた。特に重要なのは、§ 4.1 で述べたように、`JMEMSIZE` 取得にかかわる `int g5_get_jmemsize(void);` である。

4.3 コンパイル方法

GRAPE-7 のライブラリパスは、§3.2 に書いたように

```
/usr/local/g7pkg
```

である。また、インクルードファイルパス及びライブラリパスは、

```
include path : /usr/local/g7pkg/include
library path : /usr/local/g7pkg/lib
```

となっている。よってコンパイルは

```
gcc main.c -I/usr/local/g7pkg/include
-L/usr/local/g7pkg/lib -lg75 -lhib -lm
```

とする。ここでソースファイルは `main.c` とした。**ライブラリ名が異なる**ことに注意してほしい (GRAPE-5 では`-L/usr/local/grape5/lib -lg5a -lphibdma` などとしていたはずである)。

[*7] `void g5_set_xmj(int adr, int nj, double (*xj)[3], double *mj);` を使うこと。`void g5_set_jp(int adr, int nj, double *m, double (*x)[3]);` を使うとなお便利。

[*8] 近傍粒子探査機能はなくなった。したがって、これらの関数に対応するものは存在しない。

[*9] PM 法関係の関数は、現在インストールされているバージョンでは実装されていない。GRAPE-7 ライブラリ version 2.0 がインストールされたらこれらが使えるようになる。

表 3 性能比較

粒子数	GRAPE-5	GRAPE-7	比
1000	0.007 [sec]	0.0017 [sec]	∼ 4.1
10000	0.1 [sec]	0.017 [sec]	∼ 5.9
100000	8.0 [sec]	1.3 [sec]	∼ 6.1

5 性能比較

簡単なテストコードを作成し、計算時間の比較を行った。ここで用いたコードのおよその内容は付録 B に挙げておく。数字は粒子数ごとに、10 回計算を繰り返して平均をとった。なお、より丁寧なサンプルコードは GRAPE-7 のマニュアル [5] に載っている。必要があればそちらも参考にしていただきたい。

結果を表 3 に挙げる。単純な計算の範囲で 4-6 倍高速化されたことがわかる。この計算は GRAPE-7 ライブラリ version 1.3 をもちいておこなった簡単なテストである。結果は目安程度のものと思ってほしい。

6 終わりに

本稿では、国立天文台 GRAPE システム、通称 MUV(Mitaka Underground Vineyard) の GRAPE-5 ユーザが GRAPE-7 に移行するのに必要な情報をまとめた。無衝突系計算であれば、簡単な手続きを行うだけでこれまでの 10 倍近くの性能を手にすることができる。このメモと GRAPE-7 のマニュアル [5] を手に、是非新しい MUV を有効利用していただきたい。

謝辞

K&F Computing Research 社の福重俊幸氏及び川井敦氏には GRAPE-7 のライブラリ利用方法について教えていただいた。国立天文台小久保英一郎氏、東北大学の馬場淳一氏には、多忙な中原稿のチェックをしていただいた。この場を借りてお礼を申し上げる。

付録 A GRAPE-7 対応手順

1. パイプライン数を変数にする。パイプライン数は `int g5_get_number_of_pipelines(void);` を使えば取得できる。
2. `JMEMSIZE` を変数とする。`JMEMSIZE` の取得には `int g5_get_jmemsize(void);` を用いる。
3. `JMEMSIZE` およびパイプライン数が変数になったことにともない必要であれば、配列はメモリ上に動的確保するように変更する。
4. j 粒子を GRAPE に送る関数を `void`

表 1　対応表

関数名	GRAPE-7 対応
`void g5_open(void);`	○
`void g5_close(void);`	○
`int g5_get_number_of_pipelines(void);`	○
`void g5_set_range(double xmin, double xmax, double mmin);`	○
`void g5_get_range(double *xmin, double *xmax, double *mmin);`	○
`void g5_set_n(int n);`	○
`void g5_set_mj(int adr, int nj, double *mj);`	×[*7]
`void g5_set_xj(int adr, int nj, double (*xj)[3]);`	×[*7]
`void g5_set_xmj(int adr, int nj, double (*xj)[3], double *mj);`	○
`void g5_set_eps_to_all(double eps);`	○
`void g5_set_eps(int ni, double *eps);`	○
`void g5_set_h_to_all(double h);`	×[*8]
`void g5_set_h(int ni, double *h);`	×[*8]
`void g5_set_xi(int ni, double (*xi)[3]);`	○
`void g5_set_ip(int ni, double (*xi)[3], double *eps, double *h);`	×[*9]
`void g5_run(void);`	○
`void g5_get_force(int ni, double (*a)[3], double *p);`	○
`void g5_calculate_force_on_x(double (*x)[3], double (*a)[3], double *p, int ni);`	○
`int g5_read_neighbor_list(void);`	×
`int g5_get_neighbor_list(int ip, int *list);`	×
`void g5_set_cutoff_table(double (*ffunc)(double), double fcut, double fcor,double (*pfunc)(double), double pcut, double pcor);`	×
`void g5_set_cell_index_table(int nc, int *j0, int *nj);`	×
`void g5_set_eta(double eta);`	×
`void g5_set_ishift(int ishift);`	×
`void g5_set_cell_index_table(int nc, int *j0, int *nj);`	×

表 2　GRAPE-7 で新たに導入された関数のうち主要なもの

関数名
`void g5_set_jp(int adr, int nj, double *m, double (*x)[3]);`
`int g5_get_jmemsize(void);`
`void g5_set_eps2(int ni, double *eps2);`
`void g5_set_eps2_to_all(double eps2);`

g5_set_jp(int adr,int nj, double *m, double (*x)[3]); に変更する。

5. ヘッダファイルおよびライブラリパスを変更する。

```
include path : /usr/local/g7pkg/include
library path : /usr/local/g7pkg/lib
```

6. ライブラリ名に注意してコンパイルを行う。GRAPE-7 で用いるライブラリは -lg75 -lhib である。

付録 B　サンプルコード

最も簡単な重力計算ルーチンを提示する。同様の (そしてより丁寧な) サンプルコードは GRAPE-7 のマニュアル [5] に載っている。また、そちらには fortran のサンプルコードも載っている。fortran ユーザはそちらを参考にしてほしい。C 言語の最新の規格である C99 の機能を一部利用している。icc では問題なくそのまま使えるが、gcc であれば '-std=c99' というオプションを付ける必要がある。

賢明なユーザであれば、このコードと同じものを GRAPE-5 用に作成し精度について比較すると思われる。その際に答えが若干異なることがあるがこれは必ずしも誤りではない。GRAPE-7 は GRAPE-5 の互換製品であるがビットレベルで完全互換になっていないからである。当然であるが、大幅に答えが違う場合は十分に確認する必要がある。

具体的な処方は以下のサンプルの通りである。

```
#include <gp5util.h>

void calc_force(double x[][3], double m[],
   double a[][3], double pot[],
   const double eps, const int N){

 g5_open();

 int npipes = g5_get_number_of_pipelines();
 g5_set_range(-10.0,10.0,mass/128.0);
 g5_set_n(N);
 g5_set_xmj(0,N,x,m);
 g5_set_eps_to_all(eps);

 double xdummy[npipes][3];
 double adummy[npipes][3];
 double pdummy[npipes];

 for(int i=0;i<N;i+=npipes){
  int ni = MIN(npipes,N-i);

  for(int k=0;k<ni;k++){
   xdummy[k][0] = x[i+k][0];
   xdummy[k][1] = x[i+k][1];
   xdummy[k][2] = x[i+k][2];
  }

  g5_set_xi(ni,xdummy);
  g5_run();
  g5_get_force(npipes,adummy,pdummy);

  for(int k=0;k<ni;k++){
   a[i+k][0] = adummy[k][0];
   a[i+k][1] = adummy[k][1];
   a[i+k][2] = adummy[k][2];
   pot[i+k]  = pdummy[k];
  }
 }

 g5_close();
}
```

参考文献

[1] Appel, A. W. 1985, SIAM Journal on Scientific and Statistical Computing, 6, 85

[2] Barnes, J. & Hut, P. 1986, Nature, 324, 446

[3] Kawai, A., Fukushige, T., Makino, J., & Taiji, M. 2000, PASJ, 52, 659

[4] GRAPE-5 マニュアル, grape.cc.nao.ac.jp: /usr/local/grape5/g5pkg/doc/g5user.ps

[5] GRAPE-7 マニュアル, grapex.cc.nao.ac.jp: /usr/local/g7pkg/doc/g5user-j.pdf