GRAPE-4 User's Guide - Multi-Cluster version

Junichiro Makino

Department of Information Science and Graphics,
College of Arts and Sciences,
University of Tokyo,
Komaba 3–8–1, Meguro-ku, Tokyo 153, Japan
Phone : 81–3–3465–3925
Fax : 81–3–3465–3925 (same as phone)
E-mail : makino@kyohou.c.u-tokyo.ac.jp

Version 1.1 : Apr 17, 1995
Version 1.2 : June 30, 1995
Version 1.2A : May 14, 1998
Version 1.3: Sept 25, 1998

Abstract

I give the full description of the GRAPE-4 interface software package for multi-cluster system.

Contents

1 Changes **3**

1.1 Version 1.0 — Version 1.1 . . . 3
1.2 Version 1.1 — Version 1.2 . . . 3
1.3 Version 1.2 — Version 1.2A . . . 3

2 Reference Manual for subroutines **3**

2.1 DESCRIPTION . . . 5
2.2 Example . . . 7
2.3 NEIGHBOR LIST . . . 8
2.4 TIMESHARING . . . 8
2.5 SELF-INTERACTION . . . 9
2.6 LINKING . . . 9
2.6.1 Turbochannel version . . . 9
2.6.2 PCI version . . . 9
2.7 RUN-TIME SUPPORT . . . 9
2.8 BUGS . . . 9
2.9 LIMITATIONS . . . 10

3 Questions and Answers **10**

4 Update History **10**

1 Changes

1.1 Version 1.0 — Version 1.1

- Sample code for neighbor list is added (section 2.3)
- Library name is changed (section 2.6)
- ENVIRONMENT VARIABLES became necessary (section 2.7)

1.2 Version 1.1 — Version 1.2

- Support for Multi-Cluster system
- Library name is changed (section 2.6)
- ENVIRONMENT VARIABLES added
- Neighbor list function currently available only on Single-Cluster library (section 2.6)
- `H3JPDMA`, `H3JPDMA_INDIRECT` are changed to `H3MJPDMA_INDIRECT` this is to improve performance by overlapping the communication/calculation and data transfer within memory.
- `H3CALC` is now divided into two parts, `H3CALC_FIRSTHALF` and `H3CALC_LASTHALF`, in order to make it possible to overlap the calculation on host and calculation on GRAPE-4. The old `H3CALC` interface is still provided for convenience.

1.3 Version 1.2 — Version 1.2A

- Change in the configuration file location

1.4 Version 1.2A — Version 1.3

- The description of `h3mjpdma_indirect` is fixed.
- The description for the library location and bug information for PCI version added.

2 Reference Manual for subroutines

SYNOPSIS — Fortran interface

```
subroutine  h3open

subroutine h3close

function h3npipe
integer h3npipe

function h3wait
integer h3wait
```

```
subroutine h3setnboards(nb)
integer nb

function h3getnboards
integer h3getnboards

function h3getnclusters
integer h3getnclusters

subroutine h3setti(ti)
real*8 ti

subroutine h3setled(mode, iboard)
integer mode;
integer iboard;

function h3jpmax
integer h3jpmax

subroutine h3mjpdma_indirect(nj,hindex,xj,vj,aj,jj,mj,tj,mode,buffid)
integer  nj
integer  hindex(*)
real*8   xj(3,*)
real*8   vj(3,*)
real*8   aj(3,*)
real*8   jj(3,*)
real*8   mj(*)
real*8   tj(*)
integer  mode
integer  buffid

subroutine h3jpdma_start(buffid)
integer buffid

subroutine h3calc_firsthalf(nj,ni,xi,vi,eps2,h2)
integer nj
integer ni
real*8  xi(3,*)
real*8  vi(3,*)
real*8  eps2(*)
real*8  h2(*)

subroutine h3calc_lasthalf(ni,acc,jerk,pot)
integer ni
real*8  acc(3,*)
real*8  jerk(3,*)
real*8  pot(*)
```

```
subroutine h3calc(nj,ni,xi,vi,eps2,h2,acc,jerk,pot)
integer nj
integer ni
real*8  xi(3,*)
real*8  vi(3,*)
real*8  eps2(*)
real*8  h2(*)
real*8  acc(3,*)
real*8  jerk(3,*)
real*8  pot(*)

subroutine h3nbread(nb)
integer nb

function h3nblist(board,chip,nblist)
integer h3nblist
integer board
integer chip
integer nblist(*)
```

2.1 DESCRIPTION

`Grape4` is a set of simple interface routines to GRAPE-4 special-purpose N-body engine. The `grape4` routines provide minimal necessary functions to control the GRAPE-4 hardware. At present, only the FORTRAN interface is supported. Since the actual interface is written in C, it should be straightforward to a knowledgeable C programmer to use this interface.

`h3open` initializes the GRAPE-4 hardware and interface package. If someone else is using GRAPE-4, it put the process in the queue and sleep, using the lockf(2) system call (I believe).

`h3close` releases GRAPE-4 and allows other users to acquire GRAPE-3/3A.

`h3npipe` returns the number of pipelines available on the hardware.

`h3setnboards` changes the number of boards the software sees. Normally the application programmer should not use this function, unless she/he want to reduce the number of boards for test purpose.

`h3getnboards` returns the number of boards that the interface software thinks to have.

`h3getnclusters` returns the number of clusters that the interface software thinks to have.

`h3wait` is a function that waits for GRAPE-4 to finish the operation under process. The use of this function will be described in more details later.

`h3setti` sets the present time (t_i) for the predictor unit.

`h3setled` controls the operation mode of LED array on the processor board. The first argument, mode, controls the operation mode. Table 2.1 gives the summary of the mode control. **THIS FUNCTION IS NOT YET IMPLEMENTED ON MC-VERSION**

`h3jpmax` returns the maximum number of particles one can pass to subroutine `h3jpdma`

Table 1: LED mode summary

Value	meaning
24	Speedometer mode (not yet debugged)
25	Programmable mode
2n(n=0...11)	turn on led n
2n+1(n=0...11)	turn off led n

`h3mjpdma_indirect` sends the so-called j-particles data to the memory unit of GRAPE-4. `nj` is the number of particles passed. This number must not exceed the value returned by `h3jpmax`. `xj`, `vj`, `aj`, `jj`, `tj` and `mj` are position, velocity, half of the acceleration, 1/6 of the first time derivative of the acceleration, time and mass of the particle. `index` is the address to store the particle. In the case of the blockstep code, you set the indices of updated particles in this array and physical quantities of them in corresponding arrays. Note that the consecutive locations of physical arrays are used. Roughly speaking, the following will happen.

```
for(jp=0;jp<nj;jp++){
   for(k=0;k<3;k++){
      x_on_harp3[index[jp]-1][k] = xj[index[jp]-1][k];
   }
}
```

Note that the array `index` is used with fortran convention, namely, the first element is assumed to have the index one, not zero, both on host array and on GRAPE-4 memory.

The argument `mode` controls what quantities are actually send. If `mode = 1`, all of `xj`, `vj`, `aj`, `jj`, `tj` and `mj` are sent. If `mode = 0`, only `xj`, `vj`, `tj` and `mj` are sent actually sent and `aj` and `jj` are ignored. This is to save time when `aj` and `jj` are unnecessary.

The recommended use for the final argument, `bufid`, is to set it to zero. This is actually only meaningful for the implementation of the libraries with DMA support (old TurboChannel version and some of new PCI version). With the library having the DMA support, you can specify one of the buffers. In some very specific situations, you can get some speedup by using this function cleverly.

`h3mjpdma_start` actuall initiate the DMA transfer for j-particles, for the spechified buffer. This is a ummy routine for the case of non-DMA version. Usually, you call this function directly after the call to `h3mjpdma_indirect`, using the same buffer id (see example).

`h3calc` receives the positions, velocities, softenings and the neighbor sphere radii of the particles to calculate the accelerations and their time derivatives, and returns them and potentials as well. `nj` is the number of particles in the memory unit. `ni` is the number of particles on which the forces are calculated. **At present, this routine works ONLY for ni=h3npipe. ni<h3npipe would give completely wrong result.**

For historical reasons, twice the values as those given in `eps2` and `h2` are actually used.

`h3calc_firsthalf` receives the positions, velocities, softenings and the neighbor sphere radii of the particles to calculate the accelerations and their time derivatives, and starts the calculation. This function returns without waiting for GRAPE-4 to finish the calculation.

`h3calc_lasthalf` waits GRAPE-4 to finish the calculation and retrieve the force, its time derivative, and the potential for particles. The argument `ni` must be the same as its value for the last call for `h3calc_firsthalf`.

`h3nbread` transfers the content of the hardware neighbor list of GRAPE-4 to the working memory of the intereface library. At present, this subroutine **does not** return the error code even if the hardware neighbor list is overflown. This will be changed in due time. This function must be called only once after `h3calc` is called. The argument `nb` must be equal to the number of boards actually used for calculation. **THIS FUNCTION IS NOT YET IMPLEMENTED ON MC-VERSION**

`h3nblist` extracts the actual neighbor list of particle calculated in `chip` from the working memory of the inteface library prepared by `h3nbread`. The argument `nb` again must be equal to the number of boards actually used for calculation. The return value is the number of the neighbors and the `nblist` contains the particle indices (in C convention...). **THIS FUNCTION IS NOT YET IMPLEMENTED ON MC-VERSION**

2.2 Example

The following code evaluates accelerations on NI particles from other NJ particles by direct summation. Array `index` must be initialized as `index(i)=i`.

```
      subroutine calc_on_harp(nj,xj,vj,aj,jj,mj,tj,
     $     ni,xi,vi,h2,eps2,acc,jerk,pot, index, time,nnb,nbl)
      integer nj, ni,i
      real*8  xj(3,*),vj(3,*),aj(3,*),jj(3,*),xi(3,*),vi(3,*)
      real*8 mj(*), tj(*)
      real*8 h2(*),eps2(*), acc(3,*),jerk(3,*),pot(*), time
      integer jpmax, ip, nodd, index(*), npipe
      integer h3npipe, h3jpmax
      integer nnb(*), nbl(*), nbp,j
      integer h3getnboards, nboards, h3nblist
      npipe = h3npipe()
      jpmax = h3jpmax()
      nboards = h3getnboards()
      nodd = mod(nj,jpmax)
      call h3wait
      call h3mjpdma_indirect(nodd, index(1), xj(1,1),vj(1,1),aj(1,1),
     $     jj(1,1), mj(1),tj(1),1, 0)
      call h3mjpdma_start(0)
      do 30 ip=nodd+1,nj,jpmax
         call h3wait
         call h3mjpdma_indirect(jpmax, index(ip), xj(1,1),vj(1,1),
     $        aj(1,1),jj(1,1), mj(1),tj(1),1,0)
         call h3mjpdma_start(0)
 30   continue
      npipe = h3npipe()
      nodd = mod(nj,npipe)
      ip = 1
      call h3wait
      call h3setti(time)
      call h3wait
```

```
      nbp = 1
      do 40 ip=1,ni,npipe
      call h3wait
         call h3calc(nj,npipe, xi(1,ip),vi(1,ip),eps2(ip),h2(ip),
     $        acc(1,ip),jerk(1,ip),pot(ip))
         call h3nbread(nboards)
         nodd=min(ip+npipe-1,ni)
         do i = ip, nodd
            nnb(i)= h3nblist(nboards, i-ip, nbl(nbp))
            nbp = nbp + nnb(i)
         enddo
 40   continue
      end
```

2.3 NEIGHBOR LIST

The following code fragment is taken from NBODY4. This code calculate the neighbor list of particle `i`. If you wish to use the neighbor list, you should be careful of the following two problems:

a)The radii of the neighbor spheres must be the same for particles $2i$ and $2i+1$.

b)Hardware neighbor list is unlikely to available on all of 96 pipelines. Consult the author for the present situation.

```
      do KK = 3,ipmax
         h3h2(kk) = 0.0d0
      enddo
 20   continue
      h3h2(1) = h2*0.5
      h3h2(2) = h2*0.5
      npipe = h3npipe()
      nboards = h3getnboards()
      do j = 1, NPIPE
         do k = 1, 3
            xharp(k,NPIPE)=x(k,i)
         enddo
         h3eps2(j) = 1d-10
      enddo
      call h3setti(time)
      CALL H3CALC(NN,NPIPE,Xharp,vharp,h3eps2, h3h2,
     &     h3acc,h3jerk,h3pot)
      call h3wait
      call h3nbread(nboards)
      NNB = h3nblist(nboards,0,nlisti)
```

2.4 TIMESHARING

The Grape4 interface offers a rather primitive method for sharing the GRAPE-4 hardware by several programs running simultaneously. A program acquires GRAPE-4 by calling `h3open`. If someone is already using GRAPE-4, `h3open` prints some message and put the process into the sleep state. When the hardware is released, the process is waken up and `h3open` returns. A program can release

GRAPE-4 by calling `h3close`. Thus, a program occupies GRAPE-4 hardware between the calls to `h3open` and `h3close`.

In order to attain the sharing of GRAPE-4 between multiple users, therefore, programmers must write their program so that it releases GRAPE-3A as long as possible and as many times as possible.

2.5 SELF-INTERACTION

The potential energy returned includes the self-interaction. As shown in the code example above, you have to correct the returned potential on the host computer. Note that the self-potential calculated on GRAPE-4 has the accuracy of the single precision.

2.6 LINKING

2.6.1 Turbochannel version

The linker option would look like

```
-L/home2/makino/disk3src/harplibs -lharp3   -lrt -lm
```

The first is private library developed for GRAPE-4. The last two are system libraries necessary.

At present, multi-cluster library has the name `harp3_mc`

2.6.2 PCI version

The linker option would look like

```
-L/usr6/kawai/pub/lib -lg4a   -lrt -lm
```

The first is private library developed for the PCI version of GRAPE-4. The last two are system libraries necessary.

At present, multi-cluster library for PCI version does not exist.

2.7 RUN-TIME SUPPORT

The present library reads the system configuration file during initialization. This feature makes it possible to use the same executable on machines with different number of chips or boards. The configuration file has the information of defective (or nonexistent) chips on board. In addition, the number of processor boards is supplied from the environment variable. The necessary environment variables are set by sourcing the file `/home2/makino/disk3src/harplibs/set_envs`. Add in your `.cshrc` file the line

```
source /home2/makino/disk3src/harplibs/set_envs
```

DO NOT make your local copy of this file, since the content of this file may change without notice.

2.8 Known BUGS

- `H3CALC` works only with `ni = h3npipe`. WORKAROUND: use `ni = h3npipe`. This will cause slight slowdown, but would not affect the result.
- `h3mjpdma_indirect` called with `mode = 0` does not work properly. WORKAROND: use `mode=1` instead. This will cause slight slowdown, but would not affect the result.

2.9 LIMITATIONS

At present, the neighbor list gives meaningful answer only for first 48 particles. This may change in due time.

AUTHOR

Junichiro Makino

UPDATE HISTORY FOR REFERENCE MANUAL

```
12-Dec-94    Created.                        JM
17-Apr-95    Neighbor list/New linking       JM
30-Jun-95    Multi-cluster support           JM
25-Sep-98    Added description for PCI        JM
```

3 Questions and Answers

4 Update History

Version 1.0: Dec 12, 1994 — Created

Version 1.1: Apr 17, 1995 — change in the reference manual

Version 1.2: Jun 30,1995 — Multi-cluster support

Version 1.2A: May 14,1998 — File location changed

Version 1.3: Sept 25,1998 — File location changed, description for `h3mjpdma_start` changed, description for the PCI version added.