XDR-Based I/O Stream Proposal for the BBH Project

Matthew W. Choptuik
Center for Relativity, Dept of Physics
The University of Texas at Austin
Austin, TX
`matt@infeld.ph.utexas.edu`

Draft, November 20 1995

1 Motivation

The main purpose of this document is to define an `xdr`-based stream for BBH I/O. In the early phases of the BBH GC project we committed to using the `HDF` format for file I/O but never committed to a particular socket-based protocol (such as `DTM`). Nevertheless, it is clear that, at least for BBH application-development purposes, the notion of a general I/O stream (channel) which can be attached to a file or to another process (such as a visualization program) is the more appropriate concept. Whether we are streaming to a file or down the network to a graphics engine, the things (objects, data types) we are streaming are uniform lattices of function values (grid functions), labeled with a print name and (physical) temporal and spatial coordinates. There may be ancillary information about the grid function or the geometric information associated with it but it is entirely possible (and certainly simpler at this time) to assume that the analysis tools can do a good job of giving us a view of the computation using the basic quantities specified in the original GC F90 "data structure" proposal. [1]

Let me discuss a specific example to make this point more concrete. AMR introduces a lot of new "boundaries" (the extremities of a local refinement) and as anyone who has spent any time integrating finite difference equations knows, there are bound to be difficulties in the treatment of these

[1] see `http://www.npac.syr.edu/users/haupt/bbh/DATA/bbh_data.html`

regions. We are apt to want to look in detail at the interpolation of child boundary values from their parents. In 3D we will we looking at planes of values which have been interpolated from parental planes, and the values on all of these planes can be visualized very easily and directly in a system such as `Explorer`, for example, with components that are part of the basic system. The point is that `Explorer` doesn't need to know the details of our AMR structure (the whole DAGH) to provide us with the basic analysis tools we will need to get our AMR codes running—the basic geometric information (world space and time coordinates) and the function values are sufficient.

2 RNPL interface

With the above considerations in mind and proceeding on the basis that less is more when performing I/O, Robert Marsa and I defined an interface to the `HDF` routines designed to stream "GC grid functions" to one or more `HDF` files.[2] For maximum ease of use, we designed the interface so that only a single routine need be invoked to either read or write the stream. This means that the implementation of the interface has to take care of opening and closing files and maintaining an internal data base of opened files, but this is precisely the sort of thing we *want* a high-level interface to do. Thus, for example, the call

```
gft_rc = gf_write_bbox("phi",time,shape,rank,coords,data);
```

will open a file `phi.hdf`[3], if this is the first invocation on the routine with the name `"phi"`, and in any case, append the grid function `data`, labeled with the coordinates `time` and `coords` to the file. Thus, this single routine can be (and has been) inserted into a "time loop" in a typical 3D numerical relativity code to instantly provide the researcher with `HDF` output which, furthermore, can then be processed with a custom-built `Explorer` module expressly designed to expedite visualization of time-sequences of grid functions. Although we have not implemented it yet, it will be very straightforward to implement an `RPC`-based version of the `gft` routines to send data directly to `Explorer` or `AVS`.

[2]This interface is also referred to as the `bbhutil` interface: see `http://wwwrel.ph.utexas.edu:/Members/marsa/rnpl/bbhutil.ps` for man-page style documentation.

[3]the named file may already exist, in which case, depending on the setting of the `GFT_HDFAPPEND` environment variable, the file may be opened for append.

Unfortunately, we ran into significant problems with `HDF` using this approach (see the documentation), and because we are using `HDF` in such a minimal fashion—essentially for the access to `XDR` it provides—it makes considerable sense simply to replace `HDF` output with a custom `xdr` stream which can be easily implemented and verified on any of the architecture we are using or are likely to use. The `rpcgen` tool takes care of creating the low-level `xdr` routines from a `C`-like specification (`typedef` of the data to be transmitted. Read routines are slightly complicated by the fact that they may have to allocate memory for incoming data, but the `RNPL` interface already does this.

3 Structure of proposed I/O stream

The existing `RNPL` interface arguably defines the needed `typedef` since ancillary information (refinement level, component ID (which grid at this level), processor id, etc.) can be encoded in the *name* argument to the `gft` routines. In addition, as argued above, there are certainly applications where the ancillary information will not be needed. However, it is clear that the name-encoding strategy is less that ideal for all conceivable applications and there is certainly no problem in principle in transmitting the extra information. Thus the `typedef` proposed here consists of a "private" part (possibly empty) which would be self-describing in the sense that we would enumerate the "private" possibilities (`DAGH` representation, `HAMR` representation etc.) and leave the door open for essentially arbitrary ancillary information. On the other hand, the "public" part of the `typedef` will essentially be the data the `gft` routines currently transmit and is likely to be "frozen in" as soon as possible.

4 Prototype RPCGEN specification

In this section I list the input to `rpcgen` which defines an α-version of the proposed I/O stream. It should be emphasized that this is a *draft* proposal; but the final version should not be significantly more complicated. Most importantly, it was possible to define the stream and code a test application which both reads and writes test streams of grid functions (see the next section) in a few hours. The final library interface will take perhaps an order of magnitude longer, but provided we keep things simple, coding, testing and maintenance should be straightforward.

```
/* File 'bbh_gf_stream.x' ---

   rpcgen bbh_gf_stream.x

   compiles into header file 'bbh_gf_stream.h' as well as
   'bbh_gf_stream_xdr.c' which includes all XDR routines
   needed to encode/decode the BBH GF stream.           */

/* Schematic RPC description of proposed BBH I/O stream  */
/* Nov 20, 1995                                          */
/* Matthew W Choptuk, Center for Relativity, UT Austin   */

%#include <rpc/rpc.h>
%#include <stdio.h>
%#include <malloc.h>

/* Enumeration of all private implementations ... */

enum bbh_gf_stream_private {
   NONE = 0,
   DAGH = 1,
   NCSA_AMR = 2
};

/* Typedefs for each private implementation ... */


/* This info supplied by Manish Parashar ... */

struct BBH_GF_STREAM_PRIVATE_DAGH {
   string name<>;  /* Print name */
   int    rank;    /* Rank of the grid function */
   int    tstep;   /* Integer timestep */
   int    level;   /* AMR Level */
   int    comp_id; /* Component Id at the AMR level */
   int    lb<>;    /* array of dim rank holds lower bound */
   int    ub<>;    /* array of dim rank holds upper bound */
   int    shape<>; /* array of dim rank holds shape info */
   int    stride<>;/* array of dim rank holds stride in each direction */
```

```
};

/* This info culled from John Shalf's "AMR-HDF Specification ... */

struct BBH_GF_STREAM_PRIVATE_NCSA_AMR {
   int    gridID;
   int    step;
   int    rank;
   int    dims<>;
   int    origin<>;
   int    resolution<>;
   float  realTime;
};

/* Implements the union of all the private implementations ... */

union BBH_GF_STREAM_PRIVATE switch(bbh_gf_stream_private which) {
   case  NONE:
      void;
   case  DAGH:
      BBH_GF_STREAM_PRIVATE_DAGH      dagh;
   case  NCSA_AMR:
      BBH_GF_STREAM_PRIVATE_NCSA_AMR  ncsa_amr;
   default:
      void;
};

/* Enumeration of all public implementations ... the goal is to
   keep the enumeration single-valued! ... */

enum bbh_gf_stream_public {
   GFT = 0
};


/* Typedef for the basic GFT data type ... */

struct BBH_GF_STREAM_PUBLIC_GFT {
   string     pname<>;
```

```
   string     cnames<>;
   int        shape<>;
   int        n_elem;
   double     time;
   double     d<>;
};

/* Implements the union of all the public implementations ... */

union BBH_GF_STREAM_PUBLIC switch(bbh_gf_stream_public which) {
   case GFT:
      BBH_GF_STREAM_PUBLIC_GFT  gft;
   default:
      void;
};

/* Typedef for the actual BBH GF stream ... this is what an application
   program manipulates at the top level. */

struct BBH_GF_STREAM {
   BBH_GF_STREAM_PRIVATE  pr;
   BBH_GF_STREAM_PUBLIC   pu;
};
```

5 Sample Usage

The above code along with source for a test program tbbh is available via anonymous ftp to ftp://feynman.ph.utexas.edu/pub/bbh_gf_stream. Get the compressed tar file bbh_gf_stream.tar.Z from that directory and build the application using 'configure' then 'make' in the bbh_gf_stream directory created from the 'zcat bbh_gf_stream.tar.Z | tar xf -' command. Usage is

```
Usage: tbbh WRITE <file> <rank> <n> <nsteps>
       tbbh READ  <file>
```

The first form writes <nsteps> lattices, each of size <n>$^{<\mathrm{rank}>}$, to file <file>. The second form scans a file previously created by the test program (possibly on another machine) and prints diagnostics.

I have run the test application successfully on the following platforms (output from 'uname -a):

```
(a) IRIX godel 5.2 02282014 IP17 mips
(b) IRIX64 odin 6.1 07121823 IP21 mips
(c) unicos charon 8.0.4 8.0.4 CRAY J90
(d) sn4007 mario 8.0.4 viz.13 CRAY C90
(e) SunOS helmholtz 4.1.2 2 sun4c
(f) SunOS spica.cc.utexas.edu 5.3 Generic_101318-64 sun4m sparc
(g) AIX brahma 2 3 000088347000
(h) AIX r25n11 2 3 000091655700               (Cornell SP/2)
(i) OSF1 kestrel8.npac.syr.edu V3.0 347 alpha
(j) HP-UX keck3 A.09.05 A 9000/712 2013026105
```

Files created on any of these machines can be read directly on any other machine. There should be very few problems porting to any of the other architectures/UNIX variants currently in use within the Alliance since xdr is such an essential component of IPC in virtually all implementations of UNIX.

6 Acknowledgements

This work was supported by NSF grant PHY9310083 to R.A. Matzner and NSF PHY9318152 (ARPA supplemented) to the Binary Black Holes Grand Challenge Alliance.

Please forward any comments, question, complaints etc. about this proposal and the associated software to `matt@infeld.ph.utexas.edu`.