BRIO
Benchmark for paRallel I/O

Marc B.R. Joos
Service d'Astrophysique, IRFU, DSM, CEA-Saclay, France

1 Introduction

Brio is a benchmark for parallel Input/Output, using different I/O strategies commonly used in the scientific world. It is simple to use to benchmark I/O performance in your favorite architecture.

As for now, this benchmark has been successfully tried on x86 architectures with gfortran and Intel Fortran compiler and on BlueGene/Q architecture with IBM Fortran compiler.

This benchmark follows the way I/O are performed in the fixed grid version of the publicly available Ramses code[1] (Teyssier 2002, Fromang *et al.* 2006; Ramses is an adaptive mesh refinement, (magneto-)hydrodynamics code widely used in the astrophysics community).

2 I/O strategies

Five different I/O strategies can be tested with this benchmark:

- The sequential, binary POSIX I/O;
- The Parallel NetCDF library;
- The Parallel HDF5 library;
- The ADIOS library;
- The MPI-IO library.

2.1 sequential POSIX I/O

This is the classical, sequential I/O strategy. Data are written in unformatted binary Fortran. Note that this format is poorly portable. It is usually the most efficient strategy, even though it can be very bad on massively parallel computers.

[1] the code can be downloaded at `https://bitbucket.org/rteyssie/ramses`; its usage is free for non-commercial use only

2.2 Parallel NetCDF

This strategy uses the Paralel NetCDF library, which is based on the Network Common Data Form (NetCDF) library. It is a self-documented format, portable across platforms. It allows to perform high performance parallel I/O.

2.3 Parallel HDF5

This strategy uses the Parallel HDF5 library, which is the parallel version of the Hierarchical Data Format (HDF5). It is a self-documented, hierarchical, filesystem-like data format. It is portable across platforms. It allows to perform high performance parallel I/O.

2.4 ADIOS

This strategy uses the Adaptable I/O System (ADIOS), a versatile, simple and flexible library which can rely on MPI-IO, HDF5 or NetCDF4. In the current implementation of Brio, you can choose between the "XML" and "noXML" version of ADIOS; either the declarations and definitions are provided in an XML file (which is provided in the current distribution of Brio), either everything is hard-coded.

2.5 MPI-IO

This strategy uses MPI-IO, the MPI library for parallel I/O. This is highly **not** portable, producing binary files which cannot be read from one platform to the other. This is an efficient strategy however, all the other libraries being build on top of MPI-IO, adding some overheads.

3 Compiling the code

The Brio code is provided with a Makefile in which three environment variables can be defined:

- `INTERACTIVE`: to choose between the namelist interface (0) and the interactive interface (1);
- `IOTYPE`: to choose the I/O strategy. `IOTYPE` must take one of the following values: `POSIX`, `PNCDF`, `PHDF5`, `ADIOS`, `MPIIO`, `ALL`;
- `ARCH`: to choose the architecture (compiler, library paths etc.).

4 Executing the code

Some parameters can be modified at runtime, either interactively or through a namelist interface. The parameters that can be modified are:

- `xdim`, `ydim`, `zdim`: the dimensions of the local grid;
- `nx`, `ny`, `nz`: the domain decomposition, `nx` standing for the number of MPI process in the x-direction;

- `inline`: if `.false.`, the data cubes are written "as is", meaning in a highly non-contiguous way. If `.true.`; they are written along the z-direction, in a more contiguous way, improving I/O performance;
- `xml`: use only with ADIOS; if `.true.`, it will use the XML interface of ADIOS, else, it will use the noXML API.

These parameters can be modify either through the namelist `input_BRIO`, if `INTERACTIVE` is 0 in the Makefile, either through command line options (that follow the syntax `-[name of the parameter]`; for example `-nx` for `nx` parameter).

5 License

This code is distributed under the GNU General Public License (GNU/GPL).

See `http://www.gnu.org/licenses/` for more details about it.