



CONTRACT NUMBER 508830

DEISA
**DISTRIBUTED EUROPEAN INFRASTRUCTURE FOR
SUPERCOMPUTING APPLICATIONS**

European Community Sixth Framework Programme
RESEARCH INFRASTRUCTURES
Integrated Infrastructure Initiative

Grid-enabled “Group finders”

Deliverable ID: D-JRA2-2.2
Due date: October, 31, 2005
Actual delivery date: November 28, 2005
Lead contractor for this deliverable: EPCC, UK

Project start date: May 1st, 2004
Duration: 4 years

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Keywords and Abstract

Keywords:	Grid-enabling, pre- and post-processing cosmology tools, group finders, Friends-of-friends, FoF, SubFind, Virgo, GADGET, portable binary data formats, HDF5, BinX, DFDL, VOTable, FITS
Abstract:	<p>This report describes the processes involved in porting, Grid-enabling and, where necessary, parallelisation two pre- and post-processing tools employed by the Virgo Consortium. These codes are Friends-of-Friends (FoF) and SubFind. The process of Grid-enabling these tools involved the incorporation of a portable binary data format. The format employed is HDF5 and was chosen after an investigation and comparison between a number of different formats, namely HDF5, BinX, DFDL, VOTable and FITS.</p> <p>Specifically, this report is D-JRA2-2.2, and describes the comparison of different portable binary data formats and the work undertaken on FoF and SubFind in terms of the Specifications Document, D-JRA2-2.1</p>

Table of Content

1	INTRODUCTION	2
1.1	Executive Summary	2
1.2	References and Applicable Documents.....	3
1.3	List of Acronyms and Abbreviations	3
2	DESCRIPTION OF THE GROUP-FINDER TOOLS	4
2.1	FoF	4
2.2	SubFind	4
3	DESCRIPTION OF THE WORK DONE FOR FOF	5
4	DESCRIPTION OF THE WORK DONE FOR SUBFIND	5
5	COMPARISON OF PORTABLE BINARY DATA FORMATS	6
5.1	HDF5	7
5.1.1	General Description.....	7
5.1.2	Suitability	7
5.1.3	Installation	8
5.1.4	Results.....	8
5.2	BinX	8
5.2.1	General Description.....	8
5.2.2	Suitability	9
5.2.3	Installation	9
5.2.4	Results.....	9
5.3	FITS.....	10
5.3.1	General Description.....	10
5.3.2	Suitability	10
5.3.3	Installation	10
5.3.4	Performance Comparison.....	10
5.4	VOTable.....	11
5.4.1	General Description.....	11
5.4.2	Suitability	11
5.4.3	Installation	11
5.4.4	Performance Comparison.....	11
5.5	DFDL	11
5.5.1	General Description.....	11
5.5.2	Suitability	11
5.5.3	Installation	11
6	CONCLUSIONS	11

1 Introduction

1.1 Executive Summary

This project is DEISA's JRA2 – Cosmology Applications, and is a joint effort between EPCC and the UK's Virgo Consortium [1]. Funding for JRA2 comes from both DEISA and VirtU [7], where VirtU is Virgo's e-Science Virtual Universe project, funded by the UK Particle Physics and Astronomy Research Council (PPARC). VirtU started on the 1st October 2004 for 36 months and will form the foundations of the Theoretical Virtual Observatory.

All the tasks of WP2 are jointly funded by both DEISA and VirtU and, as such, all deliverables will be reported to both DEISA and VirtU.

This report is the second deliverable of WP2, D-JRA2-2.2, 'Grid-enabled "Group finders"'.

The input and output of one of Virgo's largest cosmological simulations codes, namely GADGET2 [8], are created and manipulated by a number of pre- and post-processing tools. GADGET2 has been ported to the DEISA infrastructure by JRA2's WP1 [1].

The main motivation of this work is that, once GADGET has been run on DEISA, albeit on a single platform [1], the post-processing tools can be run anywhere on the DEISA infrastructure, provided one has access to the data. This is possible due to the MC-GPFS, however, the data files must be stored in a portable data format.

The post-processing tools were to be ported to the DEISA infrastructure, Grid-enabled and parallelised where necessary. Specifically, the tools ported were Friends-of-Friends (FoF), SubFind, MergerTrees (TraceSubgroups, SplitHalos and BuildTrees), Power and Correl. This report considers only the so-called Group Finders, namely FoF and SubFind. The MergerTrees are considered in [3] whilst the Power and Correl tools are considered in [4]. The User Guide for all these tools is given in [5].

In this context, Grid-enabling means to allow the tools to utilise distributed data sets within the DEISA infrastructure, at least, and by introducing a portable, binary data format, where this format was initially chosen to be HDF5 [13], into the tool's I/O routines. It was decided that what Grid-enabling entails in practice on the DEISA infrastructure means that the associated codes must be able to access binary data files wherever they may reside on the infrastructure. This implies that the binary data files must be readable irrespective of the platform on which they were generated.

We compared the performance of a selection of portable binary data formats, namely HDF5, BinX, and FITS with the default binary formats of FoF and SubFind, and found that HDF5 had the fastest execution time of all the formats, although another portable binary data format, namely VOTable, appears to be the format of choice for Virtual Observatories, at present.

HDF5 was incorporated into the I/O routines of both FoF and SubFind. The input routines of both codes were optimised. The serial SubFind code was parallelised, whilst the FoF code was already parallel. Both FoF and SubFind have been installed, run and checked for correctness on HPCx, IDRIS and RZG, by the authors.

All work was carried out by DEISA and/or VirtU staff, except for the production of the new HDF5 and BinX input routines for FoF, the new BinX output routines for GADGET, and the performance comparison of the original binary format, HDF5 and BinX, which was all carried out by Wenjie Wei as part of his M.Sc. dissertation [20].

1.2 References and Applicable Documents

- [1] D-JRA2-1.2, Grid-enabled implementation of GADGET for metacomputing environments.
- [2] D-JRA2-2.1, Specification Document for D-JRA2-2.2, D-JRA2-2.3, D-JRA2-2.4.
- [3] D-JRA2-2.3, Grid-enabled tools to build halo merger trees.
- [4] D-JRA2-2.4, Grid-enabled tools for evaluating the basic properties of the dark matter.
- [5] D-JRA2-2.5, Documentation for D-JRA2-2.2, D-JRA2-2.3, D-JRA2-2.4.
- [6] Virgo: <http://www.virgo.dur.ac.uk>.
- [7] VirtU: <http://star-www.dur.ac.uk/~csf/virtU/virtU-final.pdf>.
- [8] GADGET: <http://www.mpa-garching.mpg.de/galform/gadget>.
- [9] FLASH: <http://flash.uchicago.edu>.
- [10] DEISA: <http://www.deisa.org>.
- [11] Barnes Hut Treecode: Barnes, J., and P. Hut, *Nature*, 324, 1986.
- [12] Monaghan, J.J., *ARA&A*, 30, 543, 1992.
- [13] HDF5: <http://hdf.ncsa.uiuc.edu/HDF5>.
- [14] BinX: <http://www.edikt.org/binx>.
- [15] EDIKT: <http://www.edikt.org>.
- [16] DFDL: <http://forge.gridforum.org/projects/dfdl-wg>.
- [17] GGF: <http://www.ggf.org>.
- [18] FITS: <http://fits.gsfc.nasa.gov>.
- [19] VOTable: <http://www.us-vo.org/VOTable/index.html>.
- [20] Wei, W., "Comparison of Portable Binary Data Formats within a Cosmological Simulation", M.Sc. Dissertation, The University of Edinburgh, 2005, available via <http://www.epcc.ed.ac.uk/msc/dissertations.htm>.

1.3 List of Acronyms and Abbreviations

FFT	F ast F ourier T ransform
GADGET	GA laxies with D ark matter and G as int E ract
MPI	M essage P assing I nterface
HDF5	H ierarchical D ata F ormat 5
BinX	B inary X ML D escription L anguage
DFDL	D ata F ormat D escription L anguage
GGF	G lobal G rid F orum
FITS	F lexible I mage T ransport S ystem
VOTable	V irtual O bservatory T able
VirtU	V irtual U niverse

2 Description of the Group-Finder Tools

The group-finder tools considered here are FoF (Friends-of-friends) and SubFind. The tools operate on binary data files generated by one of Virgo's main cosmological simulation codes, namely GADGET2.

GADGET2 is the most up-to-date version of the GADGET code. GADGET is a code used to simulate the evolution of the universe, modelling the motion of both dark matter and gas and is, essentially, an N-body simulation code that can accommodate very large values of N (particle numbers). The code calculates the gravitational forces on both dark matter and gas particles using an FFT for long range interactions and a tree algorithm (similar to the Barnes-Hut Treecode [11]) for the short range interactions. The gas component is also subject to additional hydrodynamic forces, which are calculated using Smooth Particle Hydrodynamics, or SPH (See [12] for a good overview of SPH methods)

As part of JRA2 WP1, GADGET2 was ported to the DEISA infrastructure [1].

As described in the DEISA technical annex for JRA2, WP2, we took FoF and SubFind and make them available on the DEISA infrastructure, by porting, parallelising and Grid-enabling them (see [1] for a definition of porting, parallelising and Grid-enabling).

2.1 FoF

When you run GADGET it will evolve the mass distribution, determined by the particles, to the present day, by which time clumps of particles will have formed. The purpose of FoF is to locate these clumps.

Once GADGET2 has been run, the first task is to locate particle groups that constitute dark matter halos within the check-pointed data files, or *snapshots*. These snapshots contain the coordinates, velocities, masses, etc. of all of the particles in the simulation at one instant in time. A single snapshot can be contained in a single file or across multiple files. These snapshots are then employed to determine gravitationally bound dark matter groups, the result of which is the production of a group catalogue file that enumerates the particles that belong to each of the dark matter halos identified.

The Friends-of-Friends (FoF) codes are, in fact, a collection of *Group Finder* codes. Here, we employed a code called *P-GroupFinder_Special*, which caters for both dark matter and gas. This is a parallel C/MPI code with around 3000 lines of code.

2.2 SubFind

Once the FoF group finder code has been run, it will have located the dark matter halos, identified through in the resultant *Group Catalogue* file; SubFind is then used to locate *subhalos* in the given snapshot.

The code loads the particle data and the Group Catalogue of the snapshot, and then evaluates the dark matter density for all particles which belong to a FoF group. The search of neighbours is not restricted to particles in the FoF group; SubFind considers all dark matter particles. The code examines each FoF halo in turn, including an unbinding procedure, where the gravitational potential is computed using a tree algorithm. The net

result of SubFind is that each FoF halo is decomposed into a set of substructures that are gravitationally bound. There can also be a number of particles left over ("fuzz") that are not bound to any of the subhalos.

3 Description of the work done for FoF

In its original form P-Groupfinder_Special, FoF read in all the data, via a single processor, twice: the first time only the coordinates were read in to evaluate how the particles would be distributed amongst the other processors. The required amount of memory was then allocated at these processors and the data was read in again to be distributed to the other processors.

FoF was changed to also support HDF5 as an input format. There was a minor performance problem with the code as it initially read in the particles one at a time on a single processor and then decided to which processor it corresponded. This process involved reading small amounts of data from different parts of the file because the positions, IDs and masses are not stored together.

A possible solution would have been to read all the particle data in at the beginning but this would have restricted the code to only be able analyse simulations where the data set would fit in the memory of a single processor. Clearly unsatisfactory, so large blocks of data, currently set to 10,000 particles, were read in at a time instead. This data is then distributed to the corresponding processor, also in blocks of 10,000 particles. It may be prudent to reduce this latter block value to reduce the execution time.

This input optimisation was not part of the Specifications Document [2] and has, in fact, been applied to all the tools developed by WP2.

Command line parameters are used to specify the output and input formats of the data. If it is set to 1 the native, or original binary, is used for output while a value of 2 is used to signify HDF5 output. The output is known as a *Group Catalogue*, and consists of a list of the particle ID numbers which constitute groups. The HDF5 list is identical to the binary list, and the number of particles in each group and the total number of groups found are also the same.

This new version of FoF can read single or multi-file snapshots in either HDF5 or the default native binary format.

This new version of FoF has been installed, run and checked for correctness on *HPCx*, and IBM p690+ cluster, and two of the core DEISA supercomputers, namely the platforms at IDRIS and RZG. This work was performed by the authors. This was a straightforward process, since the code uses standard language features. The fact that the I/O routines now employ HDF5 means that the binary datasets do not have to be manipulated when porting the datasets between platforms.

The work done for FoF follows the work described in the Specifications Document [2]. In addition to this, we have also optimised the input routine, as described above.

4 Description of the work done for SubFind

P-GroupFinder parallelises the code by splitting up the simulation volume into two-dimensional slabs. Each processor then works on the particles in its own slab.

As before, this works for single or multi-file snapshots.

The serial version of SubFind (FoF_Special_SubFind) can read single or multiple-file snapshots. It reads in all of the data for one snapshot by reading the corresponding snapshot file(s) in sequence. It then identifies all of the groups in the snapshot and writes out a single group catalogue. This can easily be 'parallelised' by processing each snapshot as a separate job. In this case each processor needs enough memory to store all of the particles, which limits the size of a simulation that this program can analyse to one in which the input data will fit in the memory of a single processor.

Support for HDF5 was added to the input and output routines of FoF_Special_SubFind.

The code was then parallelised in the same way as P-GroupFinder, however, the parallelisation processes was not straightforward as for the old version, as buffer particles had to be introduced to the decomposition. In addition, the I/O was parallelised as in the FoF case, see above. One processor would gather all the groups from the other processors, and write these to a single file.

This new version of SubFind can read either single or multi-file snapshots using a single processor, which then distributes the data in blocks of 10,000 to the other processors, arranged in a two-dimensional slab decomposition. These are the same techniques employed by FoF, as described above.

The code now can read and write using the HDF5 format. The code can still read in the native binary format, but this new version cannot write in the native binary format.

This task took longer than expected and, as such, the final code was not tested exhaustively. However, extensive testing demonstrated that both the HDF5 and the native binary output produced are both identical and agree with that obtained from the serial code. A pathological case may still result where, after distributing the data, one processor may end up with no particles and the code may crash. Further testing is required to rule this out.

SubFind has been installed, run and checked for correctness on *HPCx* and two of the core DEISA supercomputers, namely the platforms at IDRIS and RZG. This work was performed by the authors. This was a straightforward process, as the codes adhere to standard language features. The fact that the I/O routines now employ HDF5 means that the binary datasets did not have to be manipulated when porting the datafiles between platforms.

The work done for SubFind follows the work described in the Specifications Document [2]. Indeed, we chose the more "sophisticated parallelisation strategy", rather than the simple embarrassingly parallel method. In addition to this, we have also optimised the input routine, as described in Section 3

5 Comparison of Portable Binary Data Formats

Grid-enabling codes within the DEISA infrastructure can solely involve changing the native binary data format to a Portable Binary Data Format (PBDF). This can be the only alteration required, as the DEISA infrastructure offers a global accessible file space

across its distributed, ultimately heterogeneous platforms. Thus, if PBDFs are employed, then binary data written on one platform may be interpreted correctly on another platform, despite the fact that platforms may use different binary endianness, employ padding for alignment purposes or use different binary encodings or type representations.

To this end, we have introduced HDF5, an example of a PBDF, into FoF and SubFind, as described above. However, there are a number of PBDF schemes available. Other formats were also investigated to determine their suitability for this type of work, namely: BinX [14], DFDL [16], FITS [18] and VOTable [19] using 32 processors of *HPCx*, an IBM p690+ cluster, and 16 processors of *Lomond*, a Sun Fire 15K SMP.

This section describes our experience with each of these PBDFs in turn. The work described in Sections 5.1 and 5.2 was performed by an M.Sc. student, namely Wenjie Wei, as part of his M.Sc. dissertation [20], where more details may be found.

5.1 HDF5

The work described in this section was carried out as part of [20].

5.1.1 General Description

HDF5 is the current version of HDF (Hierarchical Data Format [13]), a general purpose library and file format for storing scientific data for exchanging data between different platforms. HDF5 uses an XML schema and saves binary data and the corresponding metadata in the same file. We cannot see metadata directly without the aid of tools. HDF5 is a well established PBDF and is widely used throughout the scientific community.

The authors of HDF5 focused on I/O performance when they developed the HDF5 Library. For instance, the HDF5 format can accommodate data in a variety of ways, such as compressed or chunked and the library is tuned and adapted to read and write data efficiently on parallel computing systems [2]. There are two optional filters, named `zlib` and `gzip`, which can be used for compressing data transparently when reading and writing.

We disabled this functionality in HDF5 library to allow for a fair comparison, as some of the other PBDFs do not support data compression on the fly. For the same reason, we also disabled the parallel I/O of the HDF5 library, as other the other PBDFs do not offer parallel I/O.

5.1.2 Suitability

HDF5 was already supported by the GADGET2 I/O system and was thus straightforward to continue using it. Some members of the Virgo Consortium already use HDF5. No other PBDF is currently used by Virgo at present.

HDF5 is a prescriptive language, in that the user must adapt how their data is stored to fit into the HDF5 schema. Indeed, there are reports that HDF5 is not the most user friendly data format and some Virgo members have suggested that, instead of forcing applications to use HDF5, each application should use their own, native binary format and employ an HDF5 converter if required.

5.1.3 Installation

An old version of HDF5 (v1.6.2) was installed on both HPCx and Lomond. GADGET2, however, required a more up-to-date version (v1.6.4). It is worth noting that different versions of HDF5 are not compatible with previous versions, and users may be required to rewrite their code.

Installing HDF5 was quite straight-forward, except the arguments of the `tr` command in the `configure` script had to be altered for Lomond.

5.1.4 Results

Using 32 processors on HPCx, and 16 processors on Lomond, we compared the speed of writing a multi-file snapshot, with 8 files in total, with 500,000 gas and dark matter particles in total, which are approximately evenly distributed across the computational domain. This corresponds to a snapshot of a smallish section of the universe just after the big bang.

There are two binary file format used in GADGET2: the default format and a slightly more complex version, which is an extension of the first. Thus, we ran 2 simulations using the default binary format and 2 simulations using the HDF5 formats. In all four cases, we read and distributed the data in blocks of 100,000.

For each test, we performed 30 runs of the simulation at different times of the day and considered the minimum, maximum, average and variance of the results, are presented in [20].

We found, to our surprise, that on both HPCx and Lomond, that HDF5 was 20% faster than the default native binary format, in writing the files, even though the HDF5 files are larger than their default binary versions, albeit by only 0.05%. The HDF developers have declared that they had optimised the I/O performance of the library and it would seem that their work has been fruitful.

For FoF, we compared the speed of reading the 8 snapshot files and, again, found that HDF5 was between twice and five times faster than the default binary, however, the binary format reads in particles one at a time, whilst the HDF5 routines read in blocks of 100,000 at a time.

For more details, see [20].

5.2 BinX

The work described in this section was carried out as part of [20].

5.2.1 General Description

BinX (Binary XML [14]) is the general name for both the BinX Language and the BinX library, and is a product of the EDKIT project [15]. In fact, the BinX Language is an individual mark-up language specified in XML. For *any* kind of binary file, we can create a separate BinX file to store associated metadata with the aid of the BinX library. This associated metadata file is written in ASCII. Thus BinX is descriptive, whilst HDF5 is prescriptive, i.e. users do not have to reformat their data in order to use BinX, whereas users must conform to the HDF5 standard.

In BinX, metadata consists of the information about the structure, physical layout, content, etc. of the binary file. Thus, we can easily transport binary data, along with associated BinX description files, between different platforms transparently without the need to modify the file itself.

In fact, we can generate the description file when we generate the binary data, or create it later for an existing binary file on the precondition that we know its structure. Almost all manipulations are then based on the description file. We can easily extract partial data from a data set from a binary file by using its corresponding BinX ASCII description file with the aid of the BinX Library or get some information rapidly from a description file without opening and manipulating the associated binary file.

There are three ways to generate a BinX file. The first one is to create a BinX file by hand. This is very convenient for creating small BinX files if users are familiar with the grammar of the BinX Language. The second way is to use external tools to generate a BinX description file using GUI. Now, BinX developers offered a tool named "BinX Editor" to edit BinX files and manipulate the data in associated binary data files. The last method is to use the BinX Library from within the software which generates the raw binary data. BinX files will be generated when the binary data files are generated. In this project, we use the last way to generate our BinX files.

5.2.2 Suitability

BinX appears to be promising, since it is descriptive and existent Virgo binary datasets do not have to be converted to ensure portability, as a BinX metadata can be created for any binary file.

However, BinX is currently only supported on Linux, Solaris and Windows. Only a subset of the full BinX library has been implemented.

5.2.3 Installation

To employ BinX, we also need to install the Apache Xerces XML parser (C++, version 2.6.0) which is necessary for the BinX library to parse the XML document. The corresponding library in form of executable binary was downloaded separately for Lomond and HPCx.

On Both Lomond and HPCx, the source code could not be compiled by `make`. The command `gmake` must be used as the syntax which `Makefile` uses is not compatible to the `make` of Lomond.

The BinX library is written in pure C++, however, we found that we could not compile GADGET2 with the C++ compiler. Thus, we had to write C wrappers to all out calls to the BinX library.

Lastly, we also found that the BinX library contained some non-standard C++ and contained some errors; however, the developers were very helpful and prompt.

5.2.4 Results

We ran the same tests as for HDF5 above. For GADGET2, which has 2 binary formats, we employed 3 different BinX formats, where, by using some of the features of BinX, we were able to reduce the amount of data written when using GADGET2's 2nd binary format. We also compared using the BinX library to write the binary file instead of using the standard method of using the C command `fwrite`.

We found that, when we ran BinX, using `fwrite` to write the binary file, the speed of writing the 8 files was comparable to the default binary method, but slower than HDF5. When we employed the BinX library to write the binary file, then the speed was a factor of 1000 slower. The two combined BinX files were up to 0.02% larger than the associated default binary files.

We then introduced BinX routines to both the input and output of FoF. We found that a bug in the BinX library did not allow FoF to read the 8 snapshot files as with the default binary format and HDF5, thus, for purposes of testing, we compared the speed of reading one single, large snapshot file.

We found that the BinX library was, in general, around 20 to 50 times slower than the default binary and HDF5 formats.

For more details, see [20].

5.3 FITS

5.3.1 General Description

FITS (Flexible Image Transport System) is the data format most widely used within astronomy for transporting, analysing and archiving scientific data files. FITS is formally endorsed by the International Astronomical Union.

Like HDF5, FITS augments the binary file itself, however, unlike HDF5, it does not use an XML schema: FITS defines its own schema. Further, FITS is also backwards compatible, unlike HDF5. HDF/FITS converters are available called `hdf2fits` and `fits2hdf`.

5.3.2 Suitability

As one of the goals of VirtU is to share data between observers and modellers and FITS is well used by the observers, then it would allow interoperability between the two groups. Indeed, there are many FITS tools available. Thus, it would appear highly suited for this project.

5.3.3 Installation

We installed FITS v3.0.0.4 on HPCx with no issues. We did not install FITS on Lomond due to time restrictions.

5.3.4 Performance Comparison

We introduced FITS into the output of GADGET2 and the input of FoF and ran the same tests as for HDF5 and BinX above.

When employing the FITS version of the GADGET2 output routines, we found FITS to be comparable to the speed of writing the default binary file, despite the FITS file being slightly larger.

However, when employing the FITS versions of the FoF input routines, we found FITS to be faster than BinX, but still around 10 to 20 times slower.

5.4 VOTable

5.4.1 General Description

VOTable (Virtual Observatory Tabledata [19]) appears to be the data format of choice for the numerous Virtual Observatory (VO) efforts around the world. Like HDF5, it uses an XML schema, however, a single VOTable document can mix ASCII and binary, so the metadata is stored within the 'binary' file, but is human readable. VOTable can also include FITS binary data files. Further, the data files can exist remotely, via a URL, which results in a format similar to BinX above.

5.4.2 Suitability

VOTable appears to be the data format of choice for any VO at this time, especially if VirtU is to interoperate with other VOs.

5.4.3 Installation

VOTable was not installed due to time restrictions; however, we note here that there are currently two Java libraries and one C++ library under development. There are also a number of VOTable tools available.

5.4.4 Performance Comparison

No performance comparison was undertaken due to time restrictions.

5.5 DFDL

5.5.1 General Description

DFDL (Data Description Format Language [16]) is similar to the BinX library, in that a binary file's metadata is stored in an associated ASCII file. The DFDL standard is currently incomplete and is being formed as part of the GGF (Global Grid Forum [16]).

5.5.2 Suitability

DFDL looks promising for the same reasons that BinX appeared promising; however, DFDL looks likely to become a data standard that is formally endorsed by GGF (the Global Grid Forum [16])

5.5.3 Installation

Currently, there is no instance of the DFDL library available.

6 Conclusions

We compared a selection of portable binary data formats, namely HDF5, BinX, and FITS, with the given binary output routines within GADGET2 and the given binary input routines within FoF, and found HDF5 had the fastest execution time of all the formats. We note that VOTable is used extensively by the majority of Virtual Observatories around the world and thus appears to be the format of choice; however, we could not investigate its performance due to time limitations.

Note that we are not excluding VirtU from interacting with other VOs by employing HDF5 rather than VOTable, since HDF5 can be considered as a single binary file and VOTable can include binary files in a single VOTable document. Further, if one converts an HDF5 file to FITS, using `hdf2fits` say, then this FITS binary file can also be included inside a single VOTable document, with the added bonus that VOTable can interoperate with FITS.

The Group Finder tools, namely FoF and SubFind, have been ported to the DEISA infrastructure, Grid-enabled and parallelised where necessary. The User Guide for these tools is given in [5].

HDF5 was incorporated into the I/O routines of both FoF and SubFind. The input routines of both codes were optimised. The serial SubFind code was parallelised, whilst the FoF code was already parallel.

The purpose of the DEISA research infrastructure is to enable scientific discovery across a broad spectrum of science and technology. The Virgo Consortium are typical of new users of the infrastructure requiring support and modest development of their software applications to take advantage of the new infrastructure. This work has been done under the auspices of the DEISA and VirtU projects and as a result a new community of scientific researchers have been brought to the infrastructure. The software applications are available for other users under the existing terms of the Virgo Consortium who are the application owners.

The objective of this work has been achieved, in that the output from GADGET cosmological simulations can now be post-processed from any of the DEISA sites, given that both FoF and SubFind have been ported to those sites.