Proceedings of The National Conference
On Undergraduate Research (NCUR) 2007
Dominican University of California
San Rafael, California
April 12 – 14, 2007

An Ontological Scheme for Specifying Time in HDF5

Ana Elena Buleu
Electrical Engineering Department, Center for Computation and Technology
Louisiana State University
Baton Rouge, LA 70803

Faculty Advisor: Dr. Gabrielle Allen
Research Advisors: Dr. Werner Benger, Ms. Shalini Venkataraman

Abstract

Hierarchical Data Format (HDF5) is a file format and an I/O library whose main strengths are a hierarchical structure, multidimensional datasets, hyperslab operations, exchangeable virtual file driver, automatic type conversions, user-definable structure types and named datatypes. At CCT, we use HDF5 as a data storage medium for large-scale scientific simulation codes (e.g. Cactus), and visualization of diverse datasets.

One open issue encountered in dealing with HDF5 is time storage. When dealing with scientific data, time is frequently a single numerical value, an amount of time units, but not a calendar date. There is no specific manner of storing time this way in HDF5, and we use instead a floating-point number. There are many accuracy drawbacks to this, such as not being able to store an offset, a time unit, a time zone, or a time scale directly associated with our number. This means that interpreting time can be cumbersome during file post-processing. Moreover, manipulating this value can lead to a loss in precision.

This paper presents my work in adding semantics associated with a floating-point number, to define time. The F5 library (an extension of HDF5) has been enhanced with functions that can read and write time to an HDF5 file. These functions can now be used in the worldwide scientific community when dealing with scientific data (usually time-dependent) stored in HDF5 files.

As a concrete example we show how the Hurricane Katrina Visualization project will benefit from this work. With the new time functions, time data in HDF5 files should be interpreted more easily by researchers.

Keywords: ontology, time, file format.

1. Introduction

1.1. motivation

Hurricane Katrina had a heavy social and economic effect on the Gulf Coast. At Louisiana State University a concerted effort was directed towards visualization research, which could be used to provide the capability to better communicate forecast models for such events. Datasets streaming from different sources had to be integrated to allow their assessment. It soon became clear that the formulation of time itself as a numerical quantity was an issue that needed improvement.

HDF5[7] is one of the file formats chosen to store scientific data at the Center for Computation & Technology (CCT). F5[8] is a library extension that provides an API based on HDF5 function calls. Whereas HDF5 provides syntax for the representation of scientific data, F5 provides a direction (the semantics) for how to formulate a certain type of scientific data.

As used in this paper, 'time semantics' refers to all the information that can be used to define a time value in a precise manner. Such information could refer to a time unit, the time zone, the calendar used, any other time value meaningful in the calculation of the current one (e.g. offset, explained later), timescale.

Figures 1, 2, and 3 respectively represent time evolutions recorded or simulated for the duration of Hurricane

Katrina. Having different origins, each of them is associated with different time notions. A common time standard of semantics has to be obtained so that they would be combined (Figure 6) in order to have a more comprehensive view of the phenomena during Katrina. This will be supported by the API developed for time semantics specification in HDF5.

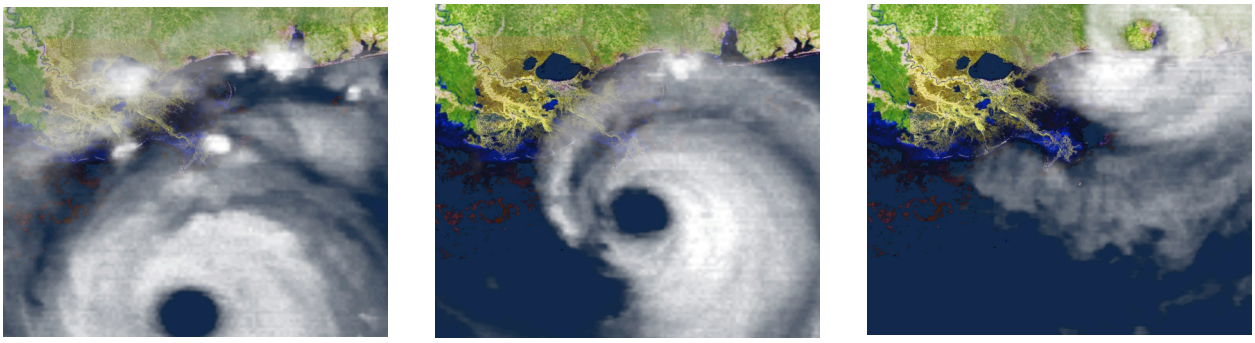

Figure 1. Cloud data recorded by satellite and shown at different time steps.

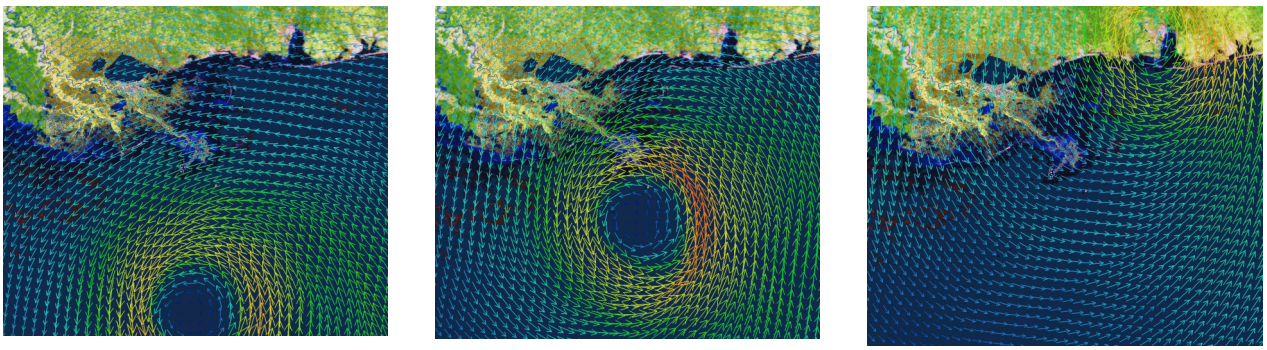

Figure 2. Atmospheric data obtained with the aid of MM5[12], a modeling software that runs simulations for generating atmospheric data and weather forecasts.

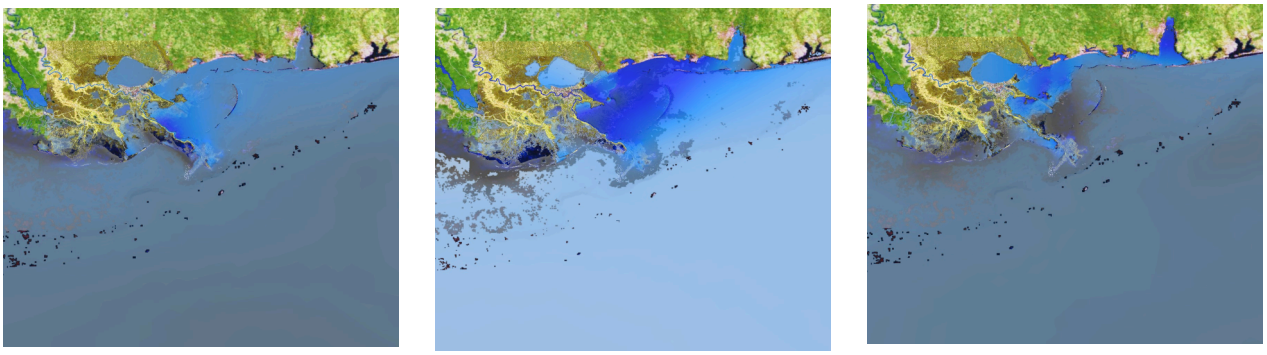

Figure 3. Sea water elevation obtained by running ADCIRC[11] Coastal Circulation and Storm Surge Model, used for analysis of surge and flooding resulting from storms.

From past experience and user feedback, we have formulated guidelines to follow in order to develop an efficient comprehensive approach to expressing time. An API created specifically for this purpose should:

1. Permit the user to specify points in time as well as intervals (in order to avoid round-off errors when subtracting two time stamps, each of which might have a different level of accuracy).
2. Allow the usage of units and scales, and conversions between different time semantics.
3. Be architecture (both software and hardware) independent (should not use calls, or structures specific to certain operating systems e.g. struct timeval – Unix specific data type).
4. Maintain precision of the original time specification (i.e. no loss in accuracy due to time unit conversions or data

type conversions).
5. Permit easy ordering of datasets within HDF5 files.
6. Have a reasonable range for specification for a given time unit (e.g. have means to avoid issues like 'the year 2038 problem', when the 32 bits used to define one memory location on most computer systems will not be enough to store time any more on Unix systems, which count time in seconds starting since January 1, 1970).

1.2. previous work

As of now, no suitable solution has been developed to specify time in the file formats that are of interest for storing scientific data in our context of research. There have been only attempts to define time as a physical quantity, but none of them is comprehensive enough to meet general needs, which have been discussed in other papers. In the geo-scientific domain, Nativi et al.[3] clearly highlighted the differences between the data models in the GIS and atmospheric sciences specifically with respect to time. The temporal scales for GIS data such as land cover tend to be in years, and orders of magnitude more than the timelines in atmospheric simulations (which are often expressed in minutes).

One of the predefined datatypes in HDF5 is "H5T_TIME". It will be implemented in the next library distribution 1.8, but it just uses the Unix time_t datatype, which has limited features.

An overview of other formats used in science has given the chance to see the current situation in time expression. Two file formats often used in science are NetCDF[5] and FITS[6]. In NetCDF there is no dedicated time type. Time-varying datasets may be represented by using the "time" dimension, but it seems that this relies very much on assumed conventions (proper naming of variables and correct specification of their attributes, such as "unit"). This approach allows for programming errors very easily, since there is no API that will catch errors of compilation.

FITS is primarily used for recording astronomical data. It has extensive support for specification of time scales and coordinates in various systems, a good start in the implementation of time semantics.

2. Theoretical background

2.1. problem description

The current state of the date and time atomic types in HDF5, as quoted from the HDF Group website, is: 'The time datatype, H5T_TIME, has not been fully implemented and is not supported. If H5T_TIME is used, the resulting data will be readable and modifiable only on the originating computing platform; it will not be portable to other platforms.' (http://www.hdfgroup.org/HDF5/doc_1.8pre/doc/H5.user/Datatypes.html)

There are two problems associated with it. The first thing noticed is that it does not agree with one of the fundamental concepts of the HDF5 file format, which is portability across platforms. This problem is addressed very well by the formulation proposed in this paper.

However, the question 'Should a formulation for time be included in the core HDF5, or should it be developed independently?' comes naturally. A pro argument for inclusion in HDF5 is the ease of access to such a time API, but this would hinder the development of more efficient formulations for addressing the same issue, given that HDF5 is powerful enough to permit the creation of an independent module dealing with this issue.

The approach proposed for the future HDF5 release does not provide many of the capabilities that are very much needed when working with time in scientific data, and gives little flexibility to disciplines which need more precision that just 32-bit precision.

2.2. different notions of time

Time is a complex topic, not only because it is an abstract notion, but also because the standards by which it is measured are set by humans in relation to natural phenomena. Different standards for measuring this quantity have been created over time. One example is calendars, some synchronized with the movements of the Sun, the Moon, or planets. There are fiscal calendars that set a fixed amount of weeks for each month, in order to facilitate comparisons between months or years. There are pragmatic calendars (based on observation, hence always accurate) and theoretical calendars (based on a strict set of rules), as well as mixed (both pragmatic and theoretical) calendars which begin as theoretical calendars, but are adjusted pragmatically when observation shows the need (e.g. the shift from the Julian to the Gregorian calendar). There have been tens of calendars used all over the world, but only some are currently used: the Gregorian calendar which is the international standard; the Hindu, Hebrew, Persian, Islamic, Chinese, Ethiopian, and Julian calendars are also used in different parts of the world. Intercalation is the insertion of

leap days, weeks, or months into calendars whenever this is needed so that the seasons or moon phases happen around the same time period every year. The Common Era is the current era, denoted as CE or AD, with correspondents for time before this era, as BCE and BC.

A time zone is a region of the earth where the same standard time is used (local time). Most geographically adjacent time zones are one hour apart, and usually express their local time as an offset from the Greenwich Mean Time. Noon is the time of the day when the sun is highest above the horizon.

Preserving the accuracy of the native time specification is important, which is why we presented a variety of formats for storing the same moments in time. However, the capability to support these formats is not as big a priority as being able to store different time scales and standards, which do not allow conversion into another time system.

Different systems of time are expressed by the Atomic Time, Universal Time (UT), the Coordinated UT, the Dynamical Time (relativistic and non-relativistic), the Geocentric Coordinate Time, the Barycentric Coordinate Time, the Sidereal Time. The difference between the Earth rotational time (UT1) and the dynamical time (TDT) is denoted by Delta T, used for predicting and keeping track of the variation of the daylight time. Daylight Saving Time and Local Civil Time are used in many parts of the world. There are computer time scales as well, like POSIX time, Microsoft Windows Time, or the Network Time Protocol.

It is important to note the difference between calendar and non-calendar time. Calendar time can be identified in a calendar and is used when recording data for which time is an important parameter, or when the event cannot be artificially reproduced at another time. Non-calendar time is merely a time coordinate, used to note the succession of the datasets, but not their 'position' on a larger timescale. It is used in numerical relativity, where the time of the simulation is meaningless in a calendar, since the simulated events are purely theoretical. Sometimes, non-calendar time is expressed by an integer counter within an iterative process.

3. Implementation

3.1. HDF5

HDF5 is a powerful file format, which is why the decision to add time semantics to it seemed worthy. One of its strengths of HDF5 is its self-describing nature, which means that the format specification and the library allows a user to find out what is stored in a file without needing additional information from the creators of the file.

An HDF5 file has a hierarchical structure, very similar to a file system, starting with a root group to which one can link groups or other named data objects. There are three kinds of named objects: groups, datasets, named datatypes. A group can be interpreted as a folder in the file system, since it can contain zero to any number of objects. A named datatype is a datatype that is attached to the file, hence will always be a part of it. Initially, one can make a copy after one of the general datatypes, or create a new one (e.g. compound datatypes), and use it, but it will not be stored in the file. Named datatypes are a powerful tool, since they allow distinguish between copies of the same data type, and they enhance the file structure. One of their important features is being able to accept attributes. Attributes can be linked only to named data objects, and are used to document an object. One attribute is made of a name and some data. In this respect, it is similar to a dataset, but it has limitations since it is created for small datasets. One has to describe the data to be stored in the attribute or in a dataset by means of a datatype, a dataspace, and a property list. The datatype is used to describe the storage layout of a single data element. The dataspace describes the layout of the elements of a multidimensional array, hence it describes the number of dimensions of a hyper-rectangle, as well as the size of each dimension. HDF5 supports up to 32 dimensions. When attaching a dataspace to an object, there can be specified a current and a maximum size for each dimension. The maximum size can be unlimited. A property list is a collection of properties used to control optional behavior for file creation, file access, dataset creation, dataset transfer (read, write), or file mounting. When performing any of these operations, there is a class of properties that can be optionally specified.

'Link' was often mentioned in the previous paragraph. The linking system is a nice feature in HDF5, which allows for its hierarchical structure, and for the creation of complex relationships between objects. Just like in Unix, it's possible to create two kinds of links. A hard link points to an object, while a soft link points to a hard link (which only tells how to get to an object), and creates an alias to it. So, deleting a hard link means deleting an object, unless there are other hard links pointing to the object, while deleting the object means that the soft link cannot be used for practical purposes any more (it only shows the way to get to the object, but it does not contain the object). However, an advantage to using symlinks is that one does not need to have created the hard link to which it points, hence the path can lead nowhere until the object is created, while in order to create a hard link, the object must be in its place

already. Figure 2 shows an example of an HDF5 file with a root group (parentless), derived groups, and, the last in the hierarchy, datasets.

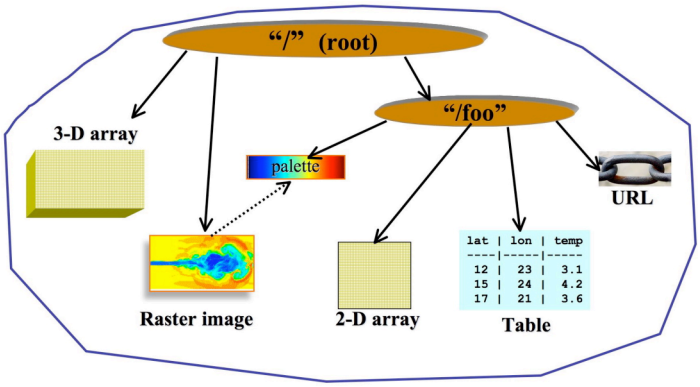

Figure 4. Structure of a possible HDF5 file

Rectangular portions of datasets (hyperslabs) can be read or written independently from the rest of the dataset. Any portion of a dataset can be accessed, if it is divided into small rectangles. Moreover, HDF5 allows for automatic data conversions between different datatypes (e.g. from double to float, from big-endian to little-endian).

The Virtual File Layer in HDF5 is an API that allows the user to create calls that would do I/O on different levels. Six drivers come with HDF5, while other already developed drivers can be added, or of course, users can create their own drivers, since this is the primary purpose of the API.

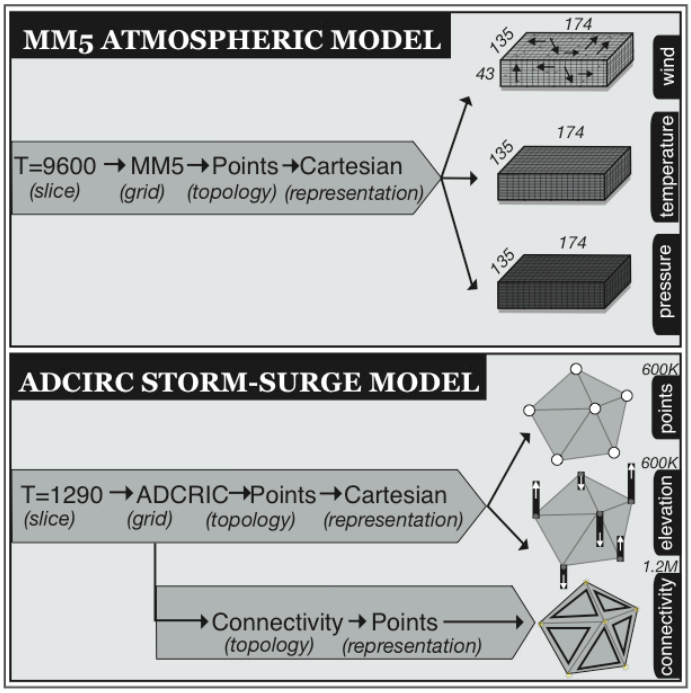

Figure 5. The structure of two F5 files that store simulation atmospheric, and surge data respectively.

3.2. F5

F5 is a C API built on top of HDF5. The main idea behind its development is to provide a data model for scientific data. It maintains the performance standards set by HDF5, while trying to complement it by adding semantics (additional information) to data stored in an HDF5 file. This is intended to help better identify 'what a number means', beyond its quantitative value.

The F5 data model is based on a mathematical concept taken from topology and differential geometry: fiber/vector bundles. In 1989, Butler and Pendly[13] came up with the idea that such a data model. The F5 data model utilizes the hierarchical structure of HDF5, but this structure becomes more specific, with six layers: Bundle → Slice → Grid → Topology → Representation → Field. Fields are the datasets. All data stored for a specific time value is grouped in a time slice. All the time slices are contained into the bundle. Slicing can be done based on any other parameter rather than time, though time is often the most important case for data systematization. The first five levels are implemented through HDF5 groups. The field level is made up of HDF5 datasets that fall into the categorization done at each level before fields. Figure 3 illustrates Katrina Visualization data in F5.

The F5 data model can be used to create an unlimited number of structures needed when representing scientific numerical data. Working with six levels of organization might seem a cumbersome job, but this hierarchy is created implicitly while the user creates a structure by calling F5 functions that describe its geometrical features.

3.3. casting time into an HDF5 type

In our formulation, time is stored as a double numerical type, but this type is cast into a named datatype in HDF5. This datatype is enriched with global attributes, which apply to all values pertaining to the data type. Each value is associated with a time slice group, hence each time slice has local (the numerical value of time, as a 64-bit double) and global time information associated with it. One advantage to using the double precision format instead of a string, time_t type (signed integer) or a compound type (e.g. the Unix struct tm, which can express calendar time) is that the values of different time steps (even if not integers) can be easily ordered, and the resulting order is meaningful in terms of time succession.

The global attributes attached to each time slice store the semantic information that allows to interpret the double value of time. This becomes crucial especially when scientists deal with multiple datasets of the same kind, hence with multiple time slices (e.g. hurricane data). In the current version of F5, the global time information is defined once in the file for all time slices, which allows consistent time semantics per file, i.e. all time information in one file has the same semantics. Future versions might allow mixing different semantics. The time unit is specified using a preliminary enum type that lists a set of predefined cases:

```
typedef enum{   UNSPECIFIED,
                UNITLESS,
                NANOSECONDS,
                MICROSECONDS,
                MILLISECONDS,
                SECONDS,
                MINUTES,
                HOURS,
                DAYS,
                YEARS,
                MEGAYEARS,
                ELECTRONVOLTS,
                METRES
            }  F5_TimeUnits;
```

Seconds are one of the seven base units in the International System of Units, as well as one of the elements in the enumeration list. Its widely used multiples (hours, days, years, megayears) and submultiples (seconds, milliseconds, microseconds, nanosecond) have also been added to it. Metres are used when time is measured in distance traveled by light, while electronvolts are used in quantum mechanics. When the creator of the file does not know with what kind of unit they are dealing, the unit is set to ‘unspecified’, and can later be changed to a meaningful unit. When there is no use to attaching a time unit to the time values (non-calendar time), time is ‘unitless’.

The structure used to store all the time information:

```
struct  _F5_TimeParameter{   double  value;
                             time_t  offset;
                             time_t  datetime;
                             F5_TimeUnits TimeUnits;
                             char    *comment;
                          };
typedef struct _F5_TimeParameter F5_TimeParameter;
```

The ‘offset’ is an amount of time units until a reference date, where the time steps in a file begin. So, in order to compute the absolute value of time for each time step, the global offset and the local time value have to be added.

This is done automatically and stored in ‘datetime’, which has been created mostly for convenience, not as a primary storage variable. It is an alias of – in general - lower precision (since the final value is an integer resulting from the addition of an integer and a double), useful with POSIX time functions (which require a time_t argument).

In the satellite atmospheric data gathered during the Hurricane Katrina (which hit New Orleans on August 29, but started tracking its course on August 15, 2005), the offset is August 15, 2005 (since we are not interested in comparing time values before this date), and the time is measured in minutes. The time slices are 30 minutes apart from each other. The first time slice is tagged with a time value of zero, while the most interesting value is usually around 22000 (corresponding to August 29).

For non-calendar time, having an offset does not make sense (e.g. black hole simulations, statistical measurement data), so the offset value will be unspecified, or, if specified, its usage is an error. In such cases, the time stored in the file is no longer calendar time, but simply a time coordinate. The ‘offset’ and the ‘value’ are initialized with zero, while the time semantics pointer has the default value of ‘unspecified’.

Currently, the API is made up of four routines:

`F5_API F5_TimeParameter*F5BnewTimeParameter(void);` - This function allocates a new F5_TimeParameter structure in memory.

`F5_API void F5deleteTimeParameter(F5_TimeParameter**);` - This function de-allocates a F5_TimeParameter structure from memory. It is passed a pointer to a pointer, and the pointer is set to zero on de-allocation, therefore it is safe to call this function multiple times on the same pointer.

`F5_API int F5setTimeUnit(hid_t file_id, const F5_TimeParameter*);` - Sets the time unit globally for the given file. The function creates a named datatype identical to the native HDF5 double type. The time unit, the offset and the comment are then attached as attributes to the named datatype. The time value is stored into an attribute attached to the time slice group. The named datatype created earlier is the declared datatype of the time value.

A return value of ‘1’ is a signal that everything went fine until the end of the routine. For errors like trying to use a zero pointer, or trying to store more than one time value at one time, or adding a dataset with different time semantics than the previous datasets added to a file (for now, time semantics consistency is required between datasets that are written in/read from the same file), the function returns ‘0’.

`F5_API int F5getTimeUnit(hid_t file_id, F5_TimeParameter*);` - This function gets the time unit information which is specified in the file. It accomplishes this by reading the attributes in which the time information is stored. On return, it behaves just like the function that was described previously.

Here is an example usage case of the API with a Hurricane Katrina dataset:

```
F5_TimeParameter* TimeParameterAddr;
TimeParameterAddr=F5BnewTimeParameter();
TimeParameterAddr->value=20000;
TimeParameterAddr->offset=mktime(/* 15 Aug 2005 */);
TimeParameterAddr->TimeUnits=MINUTES;
/*the default value for ‘TimeParameterAddr->comment’ is NULL */
F5setTimeUnit(file_id, TimeParameterAddr);
F5getTimeUnit(file_id, TimeParameterAddr);
F5deleteTimeParameter(&TimeParameterAddr);
```

4. Future work

Currently, only one set of time semantics is supported per one HDF5 file, but one of the goals kept in mind for future development is allowing conversions between these different bunches of time semantics, as well as the addition of datasets with different time semantics than other datasets in the same file. Moreover, support is needed for more means of making the time value specific, e.g. for time zones, different calendars, leap seconds, various timescales.

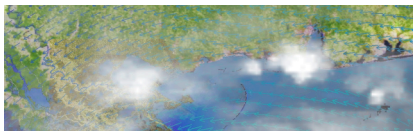
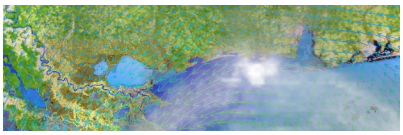
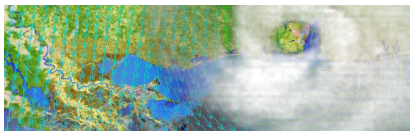

Figure 6. Combination of satellite cloud data, MM5 atmospheric data, and ADCIRC surge data, at three successive timesteps.

5. Acknowledgements

The author would like to thank Dr. Werner Benger, Ms. Shalini Venkataraman, Dr. Gabrielle Allen, Dr. Maciej Brodowicz, Dr. Edward Seidel, Mr. Marcel Ritter, Mr. Quincey Koziol, Mr. Dan Anov, Ms. Amanda Long, Mr. Enrique Pazos, and Ms. Elena Caraba for all their support while I have been working on this project, and writing the research paper. This work has been funded by the Center for Computation and Technology General Development Program.

6. References

1. Venkataraman, S., Benger, W., Long, A., Jeong, B., and Renambot, L. “Visualizing Hurricane Katrina: large data management, rendering and display challenges”. In Proceedings of the 4th international Conference on Computer Graphics and interactive Techniques in Australasia and Southeast Asia, Malaysia. pp 209-212, http://sciviz.cct.lsu.edu/papers/2006/graphite06_katrina.pdf
2. W. Benger, S. Venkataraman, A. Long, G. Allen, S. D. Beck, M. Brodowicz, J. MacLaren, E. Seidel, "Visualizing Katrina ~ Merging Computer Simulations with Observations", In Press, Springer Verlag’s Lecture Notes in Computer Science Series.
3. Nativi, S., Blumenthal, B., Habermann, T., Hertzmann, D.,Raskin, R., Caron, J., Domenico, B., Ho, Y., Weber, J.: Differences among the data models used by the geographic information systems and atmospheric science communities. In: Proceedings American Meteorological Society - 20th Interactive Image Processing Systems Conference. (2004)
4. W. Benger, "Visualization of General Relativistic Tensor Fields via a Fiber", PhD thesis, Free University, Berlin, 2004.
5. "netCDF Time Discussions by thread, http://www.unidata.ucar.edu/software/netcdf/time/
6. R. J. Hanisch, A. Farris, E. W. Greisen, W. D. Pence, B. M. Schlesinger, P. J. Teuben, R.W.Thompson, A. Warnock III, "Definition of the Flexible Image Transport System (FITS)", http://www.aanda.org/index.php?option=com_base_ora&url=articles/aa/full/2001/34/aah2901/aah2901.right.html&access=standard&Itemid=81#s:tsys
7. "HDF5 Homepage", http://www.hdfgroup.org/HDF5/
8. “The Fiber Bundle HDF5 Library”, http://www.fiberbundle.net
9. "Time Scales", http://www.ucolick.org/~sla/leapsecs/timescales.html
10. "Time", http://en.wikipedia.org/wiki/Time
11. “ADCIRC”, http://www.adcirc.org/
12. “MM5 Community Model Homepage”, http://www.mmm.ucar.edu/mm5/overview.html
13. David M. Butler and M. H. Pendley, “A visualization model based on the mathematics of fiber bundles”, Computers in Physics 3 (1989), no. 5, 45–51.