



IST-2001-32133

GridLab - A Grid Application Toolkit and Testbed

## Deliverable D8.8 - Parallel dynamic data filters

---

Author(s):	Felix Hupfeld, Andrei Hutanu, Thorsten Schütt, and Brygg Ullmer
Document Filename:	
Work package:	WP 8 - Data Handling and Visualization
Partner(s):	
Lead Partner:	ZIB
Config ID:	GridLab-08-DATA-0007-M30
Document classification:	IST

---

**Abstract:** This document provides a discussion of WP8's activities and results regarding parallel dynamic data filters.



## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>GridFTP</b>	<b>2</b>
<b>3</b>	<b>HDF5 filter</b>	<b>3</b>
3.1	Meta Data Filtering . . . . .	3
3.2	Data Set Reading and Subsampling . . . . .	3
3.3	Client-side . . . . .	4
<b>4</b>	<b>FALLS filter</b>	<b>4</b>

## 1 Introduction

In the Annex document, WP8 was assigned commitments for the design and implementation of (parallel) dynamic data filters, utilizing the complete scope of now existing data flow and data management techniques. These filters were intended to be rendered accessible through a generic API, and to form the most generic part of the techniques of WP8.

One of the core GridLab objectives lies in the embracement and extension, to the extent possible, of existing standards and technologies developed within the Grid community. Toward this end, WP8 has invested in and build upon the GridFTP protocol defined by the GGF Forum, and specifically the software implementation developed by the Globus group.

This decision has several important consequences. GridFTP supports parallel processing; parallel streams; and a modular “plug-in” approach, facilitating extensibility through (e.g.) new data filter mechanisms. Thus, the identification and adoption of GridFTP provides GridLab both with mechanisms for parallelism, as well as a path for implementing performance-enhancing data filters.

WP8 has focused its development activities on this later aspect: the research, design, and implementation of efficient remote data filters which are useful to our collaboration partners. We discuss these efforts in three sections. First, we briefly introduce GridFTP, including its mechanisms for parallelism and plug-in extensibility.

Second, we discuss our GridFTP plugin providing an HDF5 filter. This activity has been published in [1] and [2]. HDF5 is a widely used data format in the scientific community, and one which is heavily used by our astrophysicist collaborators. We have implemented mechanisms that parse HDF5 files; efficiently aggregate and transport HDF5 meta-data; and select and transfer subsections of these files.

In the final section, we discuss a technology known as “FALLS” (FAMILY of Line Segments). FALLS provide a generic mechanism for describing patterns across arbitrary binary files. Seen from the perspective of GridLab, pitfalls provide a generic mechanism for dramatically accelerating the selective transfer of data file subsets from large binary files. This work was introduced in D8.5; here, we briefly summarize its relevance to our D8.8 commitment.

Both of these plugins offer dynamically-selectable filters which can accelerate remote file access by several orders of magnitude, when compared with naive access methods.

## 2 GridFTP

The GridFTP protocol plays a central role in our data access schema. This approach gives us a number of advantages when compared with approaches implemented on top of custom or proprietary protocols:

1. GridFTP supports server side data processing, which we utilize for data filtering.
2. The GridFTP protocol, as an extension to the standard FTP protocol, is well-known and embraced by a relatively broad user community.
3. GridFTP allows the incorporation of standard servers for solutions with limited functionality.
4. The GridFTP infrastructure takes care of:
  - establishing the data connection;
  - ensuring authentication and authorization;

- invoking the data filter plugin; and
- performing the data transfer;

Significantly, the GridFTP protocol enables support for adding custom commands for server side data processing. Specifically, the plugins offered by a server define sets of ERET and ESTO parameters that correspond to the data filter module implemented by the plugin. The extended store (ESTO) and extended retrieve (ERET) commands of the GridFTP protocol are defined as following:

```
ESTO <module_name>=<modules_parms> <filename>
ERET <module_name>=<modules_parms> <filename>
```

`module_name` is a server-specific string representing the name of the module to be used. The second string (`module_parms`) is module specific and defines the operation to be performed by the module. The last parameter (`filename`) specifies the file to be processed, which can be any file that can be processed by the given module. In our case, filenames should specify HDF5 files for the HDF5 module, and any binary file for the FALLS module.

### 3 HDF5 filter

We use the GridFTP protocol to define two operations that can be applied to HDF5 files: one for meta data filtering, and a second for data access.

#### 3.1 Meta Data Filtering

The first operation of the HDF5 filter is parsing meta data from the HDF5 file. This is achieved by creating a filtered copy of the original file. Toward this end, the module reads and parses the original file, and writes the meta data information to a copy of the file. When copying (writing) a data set, we use the HDF5 filter interface and apply a filter to the original files data set. The resulting file contains only the meta data, without containing any of the actual dataset content. All other information – e.g., the hierarchy (groups), attributes, and data set information (name, data type and data space) – is preserved.

The generated file is transferred to the requesting client using GridFTP. The ERET command for requesting the meta data file is:

```
ERET HDF5="METADATA" <filename>
```

#### 3.2 Data Set Reading and Subsampling

The second operation performs data selection and filtering. The client can choose to read an entire data set, or a portion of the data set. The HDF5 data sets logically group the actual data within multidimensional arrays named "data spaces." The model we use to specify a portion from a data set is based on the HDF5 "hyperslab" model. A hyperslab describes either a contiguous collection of points, or a regular pattern of points or blocks in the data spaces. A hyperslab is specified by four parameters:

- origin: the starting location;
- size: the number of elements (or blocks) to select along each dimension;

- stride: the number of elements to separate each element (or block) to be selected; and
- block: the size of each block selected from the data set.

All of these parameters are one-dimensional lists, with lengths equal to the number of dimensions of the data set. The elements of these lists specify data array lengths or offsets for corresponding dimensions of the data arrays.

Our mechanism for specifying the hyperslab coordinates takes the following form:

```
ERET HDF5="BLOCK:NAME=<datasetname>;\  
          DIMENSIONS=<n>;\  
          ORIGIN=<orig0>,<orig1>,...,<orign-1>;\  
          SIZE=<size0>,<size1>,...,<sizen-1>;\  
          SAMPLING=<sampling0>,<sampling1>,...,<samplingn-1>;\  
          BLOCK=<block0>,<block1>,...,<blockn-1>"  
          <filename>
```

`datasetname` is the fully qualified name (including the path to the data set) of the data set from which data should be read; `orig0` to `orign-1` are the coordinates of the first block to be selected from the data set; `size0` to `sizen-1` are the number of blocks to be selected in each dimension; `sampling0` to `samplingn-1` represent the distance between two selected block for each dimension (in data elements) and `block0` to `blockn-1` specify the dimensions (in data elements) of a selection block.

This request is sent to the server. The server opens the file *filename*, opens the given data set, and reads the portion of the file specified by the given parameters. If the selection is too large, it will be dynamically split in multiple smaller selections on the server side in order to fit in the main memory. Next, the retrieved data is sent via the GridFTP connection to the client. If the selection was internally split on the server side this will not be visible to the client.

### 3.3 Client-side

The HDF5 library provides a mechanism for supporting external I/O drivers named Virtual File Drivers. This allows the simple integration of various I/O mechanisms in the HDF5 library. In this way, the access to the data uses the same interface independent of the data locality (data stored locally or remotely). With a small modification in the HDF5 pipeline, we were able to extend the capabilities of the virtual file driver concept in order to make efficient use of our high-level filtering methods. With this, our users can now employ the full power of the HDF5 API, while enjoying the full performance offered by our remote HDF5 filter.

## 4 FALLS filter

For the HDF5 handling we added file format specific code to the GridFTP server. This gives us the best performance but on the other hand we increased the administration overhead by requiring the HDF5 plugin on the server side.

We also wanted to have a remote partial file access framework which uses an file format independent plugin on the server side. Tests using the facilities provided by GridFTP showed inferior performance. The network traffic was dominated by the control flow and not by the data flow. Therefore we searched for a powerful language for describing subsets of files.

FALLS ( FAmiLy of Line Segments) were originally introduced for transformations in parallel computing. There is also a parallel filesystem which uses FALLS to describe the file layout. They

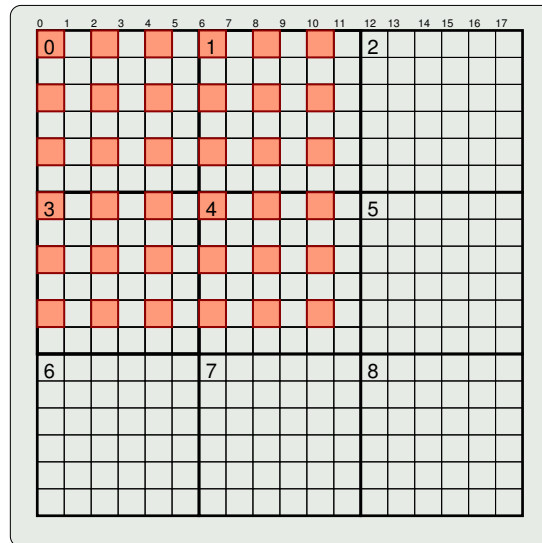


Figure 1: Regular subset of a two dimensional matrix

can be used to describe regular subsets of arrays with a very compact syntax. E.g. the following FALLS describe the highlighted elements of the matrix in Fig 1:  $(0, 17, 36, 6, (0, 0, 2, 6))$ .

The knowledge about the file format is not needed on the server side anymore. We developed drivers several file formats based on FALLS. Depending on the parameters the performance was several magnitudes better than with stock GridFTP. For in depth discussion including performance numbers please see [3].

## References

- [1] H.-C. Hege, A. Hutanu, R. Kähler, A. Merzky, T. Radke, E. Seidel, and B. Ullmer. Progressive retrieval and hierarchical visualization of large remote data. In *Proc. Workshop on Adaptive Grid Middleware*, pages 60–72, September 2003.
- [2] S. Prohaska, A. Hutanu, R. Kähler, and H.-C. Hege. Interactive exploration of large remote micro-ct scans. *IEEE Visualization '04* (accepted).
- [3] T. Schütt, A. Merzky, A. Hutanu, and F. Schintke. Remote partial file access using compact pattern descriptions. In *IEEE/ACM Intl. Symp. on Cluster Computing and the Grid - CCGrid2004*. IEEE Computer Society, April 2004.