GridFS
An Interface to Grid Replica Location and File Storage Services

András Nagy

Andras.Nagy@cern.ch

2002-08-29

Goals

Enable any application to access "files in the Grid" efficiently.

- *Any application.* Make Grid resources accessible as regular files in the Unix file system; no changes to existing programs will be required.
- *Files in the Grid.* Combine optimized replica location and file access services, so that specifying a single logical file name would automagically locate the nearest replica and access that transparently.
- *Efficiently.* Do not download the whole file if only a small fraction of it is accessed.

Mechanism

User space virtual file system:

- Plug a module into the Linux kernel which forwards relevant system calls (`open`, `read`, etc) to a user space daemon which handles them.
- As the daemon runs in user space, it can use the various shared libraries for replica location and file access, which provide the required funcionality.

What about SlashGrid?

Yes, this is the same idea. Unfortunately, SlashGrid uses the Coda kernel module, which does not allow partial access to files. (Recall the last requirement on slide *Goals*.) This problem is currently being worked around in SlashGrid, however, I don't believe that the proposed solution is feasible.

What about Condor's Pluggable File System?

Condor's PFS is a preloaded shared library. As such, it is tied to a specific version of glibc, and even worse, for full funcionality, glibc must be patched. Furthermore, by it's nature, it does not work with statically linked programs or programs linked to an older version of libc. As our goal is to support *any* application, including legacy ones, which cannot be recompiled for whatever reason, this approach is not feasible.

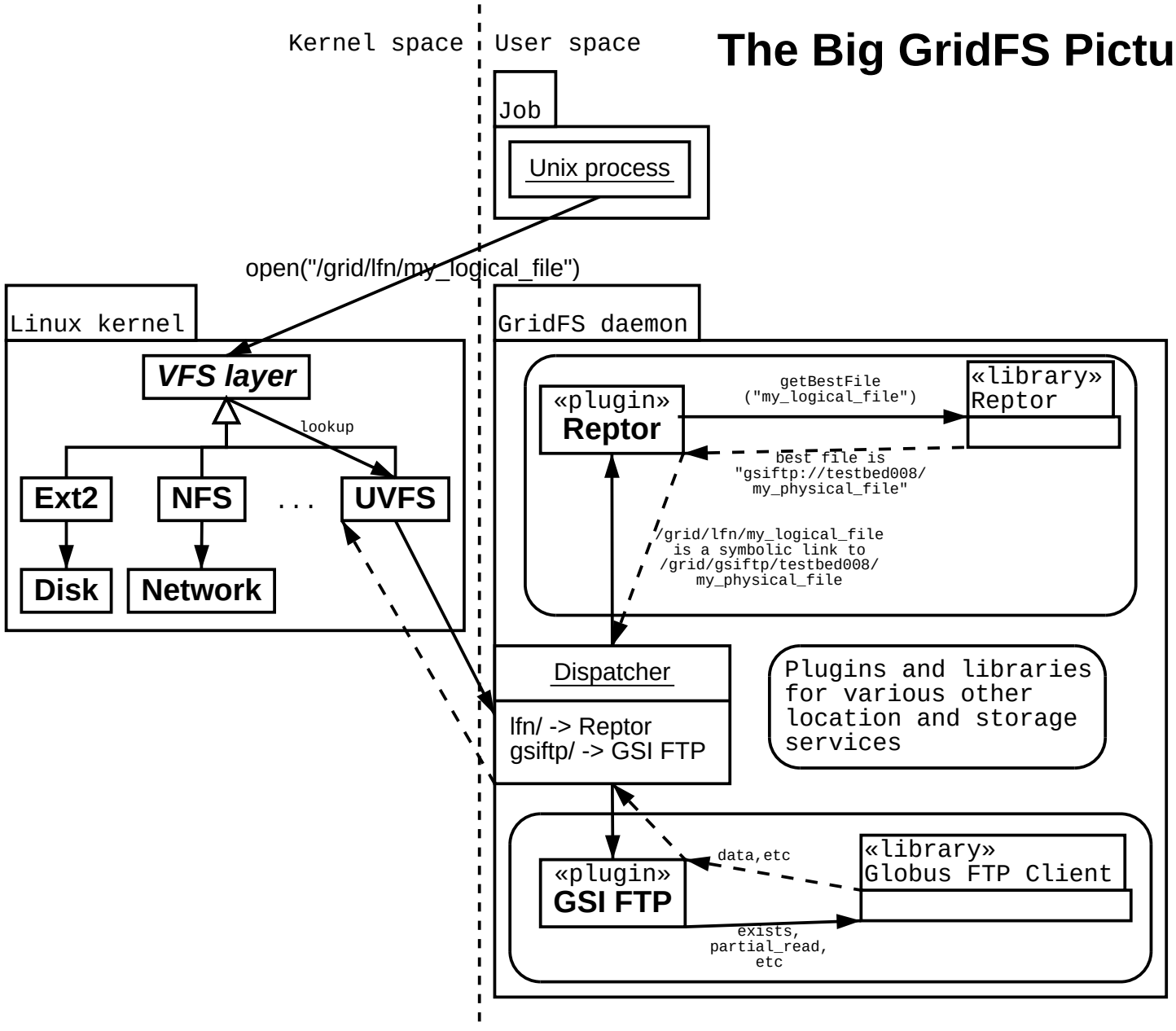
Prototype implementation

In the prototype, the following compontents are being used:

- *Kernel module:* UVFS by Britt Park.
- *Optimized replica location service:* Reptor from EDG WP2.
- *File access service:* Globus FTP Client from Globus.

As the daemon is designed with abstraction in mind, further services (plugins) should be easy to add.

The Big GridFS Picture



Current status: Core

Done in general, cosmetics needed. Noteworthy points:

- *Kernel module.* Various tweaks were needed to make it suitable for our situtaion. This took more time than I expected, as I had to understand almost every bits and pieces of UVFS and the Linux VFS layer, while documentation on both of them was rather weak.

Current status: Core (II.)

- *User space daemon.* Designed an abstract interface, described in C++, for both the kernel–daemon and daemon–plugin communication. File system objects (inode, file, etc) are mapped to C++ objects. A plugin inherits from the generic classes to implement it's own file semantics.

Current status: Globus FTP Client

Implemented read-only access. Unfortunately, several bugs in the Globus FTP Client library make the whole thing unusable. Namely:

- *Implicit credential acquisition.* The GridFS daemon operates on behalf of another user, therefore it has to acquire credentials for that user, before doing GSI FTP operations. This works fine, however, the Globus FTP Client library, even if provided with a valid GSSAPI credential, tries to acquire one using the default method, and will obviously fail. Therefore, currently only a single user can access the file system provided by the daemon, and the daemon has to be run as that user.

Current status: Globus FTP Client (II.)

- *Crash upon ‘connection refused’.* When trying to open a connection to a server where no GSI FTP daemon is running, an assertion inside Globus FTP Client fails, and crashes the whole GridFS daemon. This introduces the possibility for a trivial denial-of-service attack.

Current status: Globus FTP Client (III.)

- *No caching.* Altough it is supposed to, Globus FTP Client does not cache FTP control or data connections; for *each* operation a new connecion is opened and closed. As data is read in memory-page sized blocks (usually 4K), the result is zero performance.

These bugs are currently being reported to the Globus team.

Current status: Reptor

To be done. Shouldn't be difficult at all, as only the lookup operation has to be implemented, which must make a single getBestFile() call and return the result as a symbolic link.

Availability

GridFS code will be imported into the EDG CVS tree soon.

Thank you for your attention!