Update on GridFTP-Lite

Bill Allcock, ANL

BNL Network Research PI Meeting

29 September, 2005

the globus alliance
www.globus.org

Before we start...

- This is to remind me to start the build... ☺
- From a recent CERN press release (link):

Initially, PHENIX had planned to transfer the polarized proton-proton data by physically transporting tape cartridges to CCJ. During the early part of the run, however, it was found that network transfer rates of 700-750 Mbits/s could be achieved. A dedicated network path was established from the PHENIX counting house to the BNL perimeter network, and the tape option became a fall-back solution. In the end, not a single tape was shipped.

The principal tool used for the transfer was GridFtp, which proved to be very stable. Brookhaven has a high-speed connection (OC48) to ESNET, which is connected to a transpacific line (10 Gbit/s) served by SINET in Japan. Apart from two half-day outages of ESNET, the transfers continued around the clock for the entire 11 week run.

Approximately 270 TB of data (representing 6.8 billion polarized proton-proton collisions) were transferred to CCJ. After a few days of fine-tuning the transfer parameters, the transfers became part of the regular data-handling operation of the PHENIX shift crews, requiring experts to intervene only occasionally.

What is GridFTP?

- A secure, robust, fast, efficient, standards based, widely accepted data transfer protocol
- A Protocol
  - Multiple independent implementations can interoperate
    - This works. Both the Condor Project at Uwis and Fermi Lab have home grown servers that work with ours.
    - Lots of people have developed clients independent of the Globus Project.
- Globus supplies a reference implementation:
  - Server
  - Client tools (globus-url-copy)
  - Development Libraries

the globus alliance
www.globus.org

GridFTP: The Protocol

- FTP protocol is defined by several IETF RFCs
- Start with most commonly used subset
  - Standard FTP: get/put etc., 3rd-party transfer
- Implement standard but often unused features
  - GSS binding, extended directory listing, simple restart
- Extend in various ways, while preserving interoperability with existing servers
  - Striped/parallel data channels, partial file, automatic & manual TCP buffer setting, progress monitoring, extended restart

GridFTP: The Protocol (cont)

- Existing standards
  - RFC 959: File Transfer Protocol
  - RFC 2228: FTP Security Extensions
  - RFC 2389: Feature Negotiation for the File Transfer Protocol
  - Draft: FTP Extensions
  - GridFTP: Protocol Extensions to FTP for the Grid
    - Grid Forum Recommendation
    - GFD.20
    - http://www.ggf.org/documents/GWD-R/GFD-R.020.pdf

wuftpd based GridFTP

Functionality prior to GT3.2

- Security
- Reliability / Restart
- Parallel Streams
- Third Party Transfers
- Manual TCP Buffer Size
- Partial File Transfer
- Large File Support
- Data Channel Caching
- Integrated Instrumentation
- De facto standard on the Grid

New Functionality in 3.2

- Server Improvements
  - Structured File Info
    - MLST, MLSD
  - checksum support
  - chmod support (client)
- globus-url-copy changes
  - File globbing support
  - Recursive dir moves
  - RFC 1738 support
  - Control of restart
  - Control of DC security

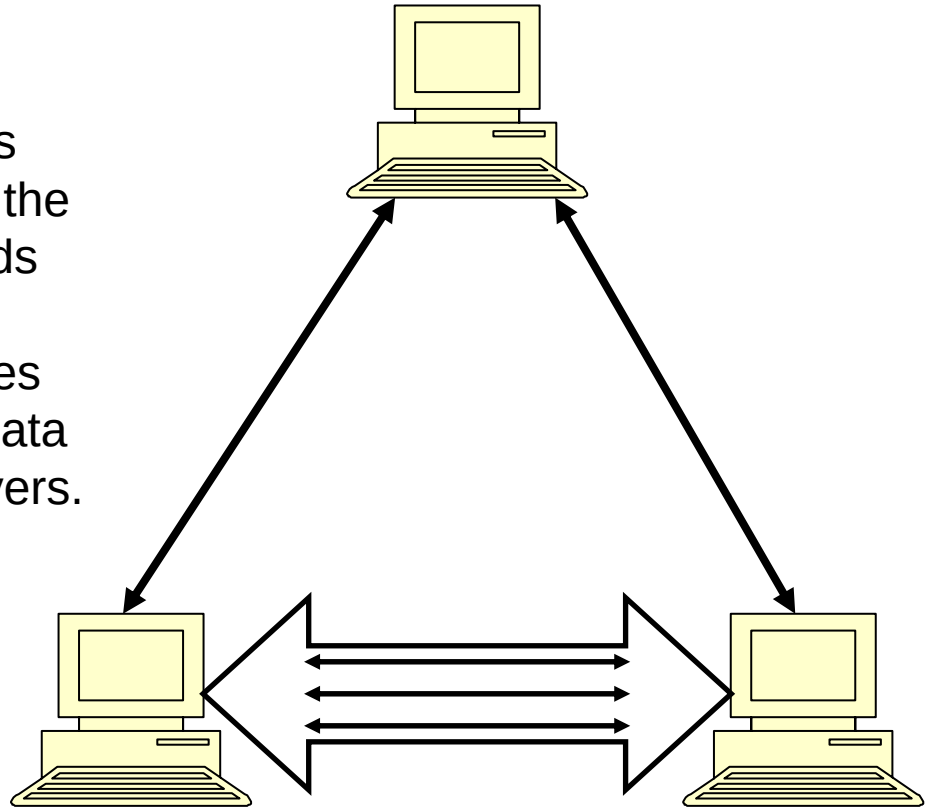
the globus alliance
www.globus.org

New GT4 GridFTP Implementation

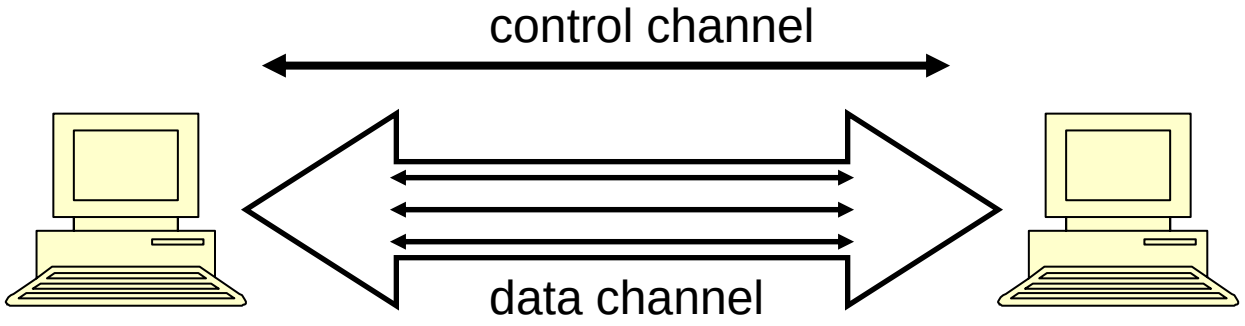
- NOT based on wuftpd
- 100% Globus code. No licensing issues.
- Striping support has been added
- Has IPV6 support included (EPRT, EPSV), but we have limited environment for testing.
- Extremely modular to allow integration with a variety of data sources (files, mass stores, etc.)
- Based on XIO
- wuftpd specific functionality, such as virtual domains, will NOT be present

3rd Party Transfer

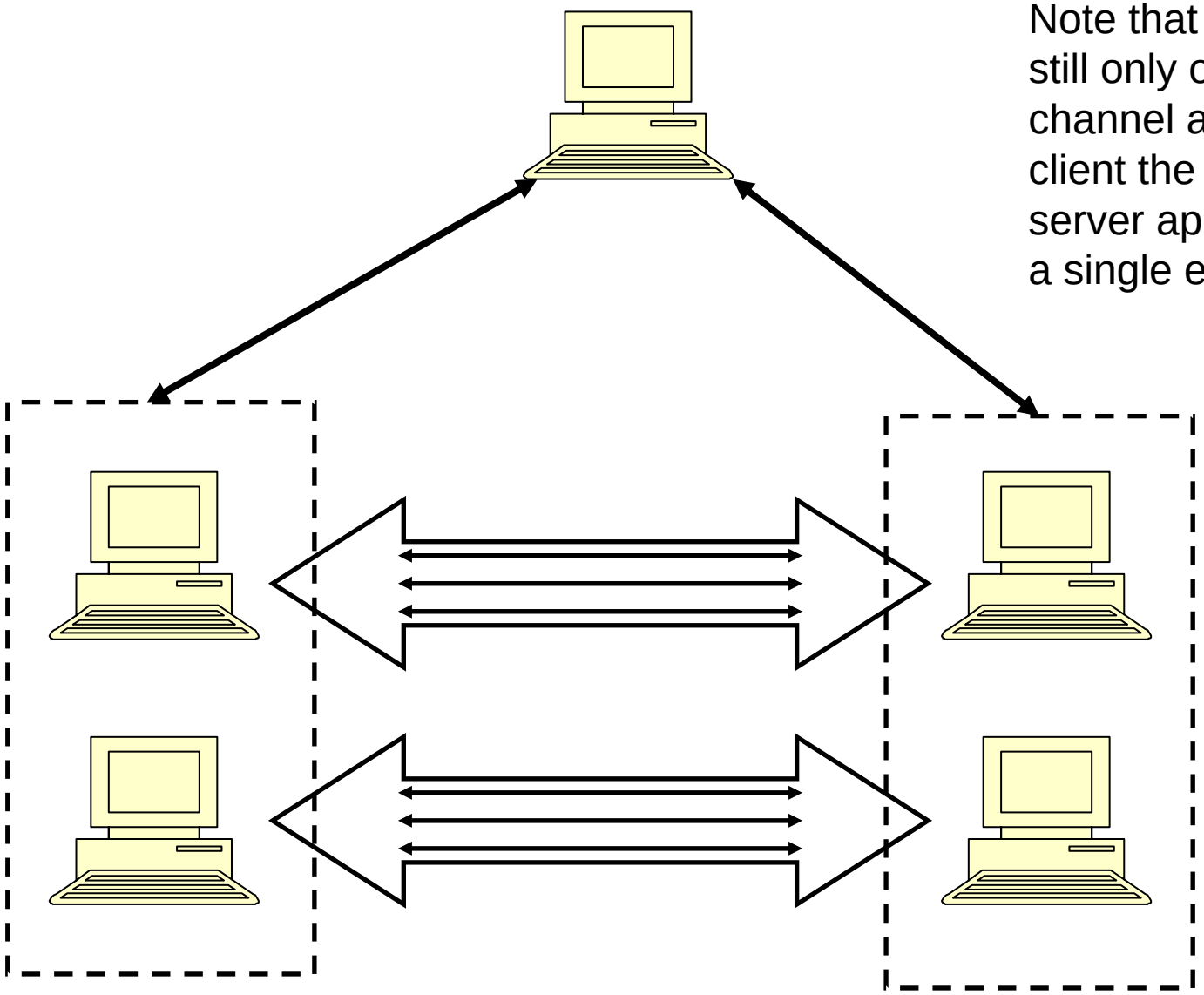
client sends commands and receives responses over the control channel, the server receives commands and sends reponses. In this case, the client orchestrates the transfer, but the does NOT take part in the transfer. The data moves directly between the two servers.



Parallelism



Striping



Note that there is still only one control channel and to the client the striped server appears as a single entity

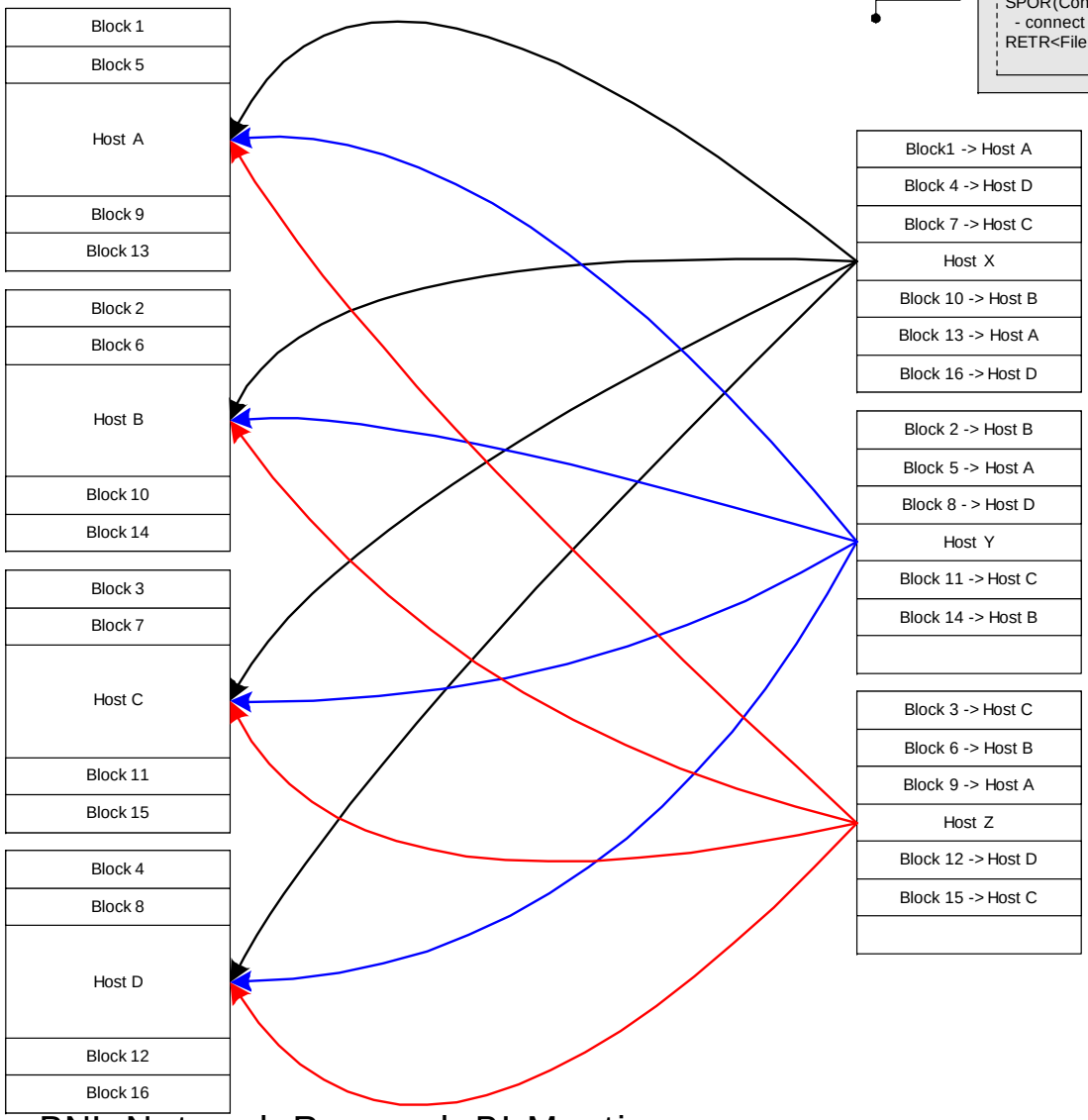
the globus alliance
www.globus.org

18-Nov-03

GridFTP Striped Transfer

MODE E
SPAS (Listen)
- returns list of host:port pairs
STOR<FileName>

MODE E
SPOR(Connect)
- connect to the host-port pairs
RETR<FileName>



the globus alliance
www.globus.org

Goals of the Project

- Make GridFTP a viable option for the network research community
  - Ease of use
    - improved packaging and build interface
    - highly portable / binaries available for many systems
    - alternative security options
  - Modular to ease modification
    - XIO drivers abstract underlying transport protocols
    - Data Storage Interface abstracts the underlying data source/sync
  - Specific Features

Ease of Use

Improved Packaging / Build

- GPT (Grid Packaging Toolkit) is still there under the covers
- However, we have hidden it under a fairly complicated (for us) makefile.
- To you, it is now the familiar configure, make, make install
- You *ARE* downloading the entire Globus source, but you are only building the pieces necessary for GridFTP
- It may complain if you don't have Java, but you can ignore that.

How to Install GridFTP

- download the latest all source installer from the Globus toolkit web page http://www.globus.org/toolkit
- tar –zxf gt4.0.1-all-source-installer.tar.gz
- cd gt4.0.1-all-source-installer
- export GLOBUS_LOCATION=“install path”
- ./configure prefix=$GLOBUS_LOCATION
- make gridftp postinstall
- source $GLOBUS_LOCATION/etc/globus-user-env.sh
- $GLOBUS_LOCATION/sbin/globus-gridftp-server –p <port> -aa –allow-anonymous-names anon
- globus-url-copy ftp://anon@localhost:port/foo file:///bar
- Server configuration documentation

So, you want to build faster…

- This is NOT officially supported. It usually works, but the parallel builds fail sometimes. Use it if you like, but don't complain if it doesn't work.
- Some packages can be built parallel, some can't
- By default, everything is built sequentially for safety.
- If you have a hot machine (especially fast disk), try this:
- make gpt globus_core globus_core-thr
- make gridftp –j# (# is parallelism, I use 8)
- This is what I used for the live build
  - at least that was the plan ☺

the globus alliance
www.globus.org

Highly Portable / Binaries

- We don't have an official list of supported platforms, because ALL support is best effort
- We build on 3 Linux variants every night
- We build on a broad range of platforms for every release.
- 64 bit is not a problem
- If we don't run on your platform try it, if you have problems we will help you as best as we can
- If you make a platform available to us, we *may* be able to build on it before releases.
- Release notes / binary download page

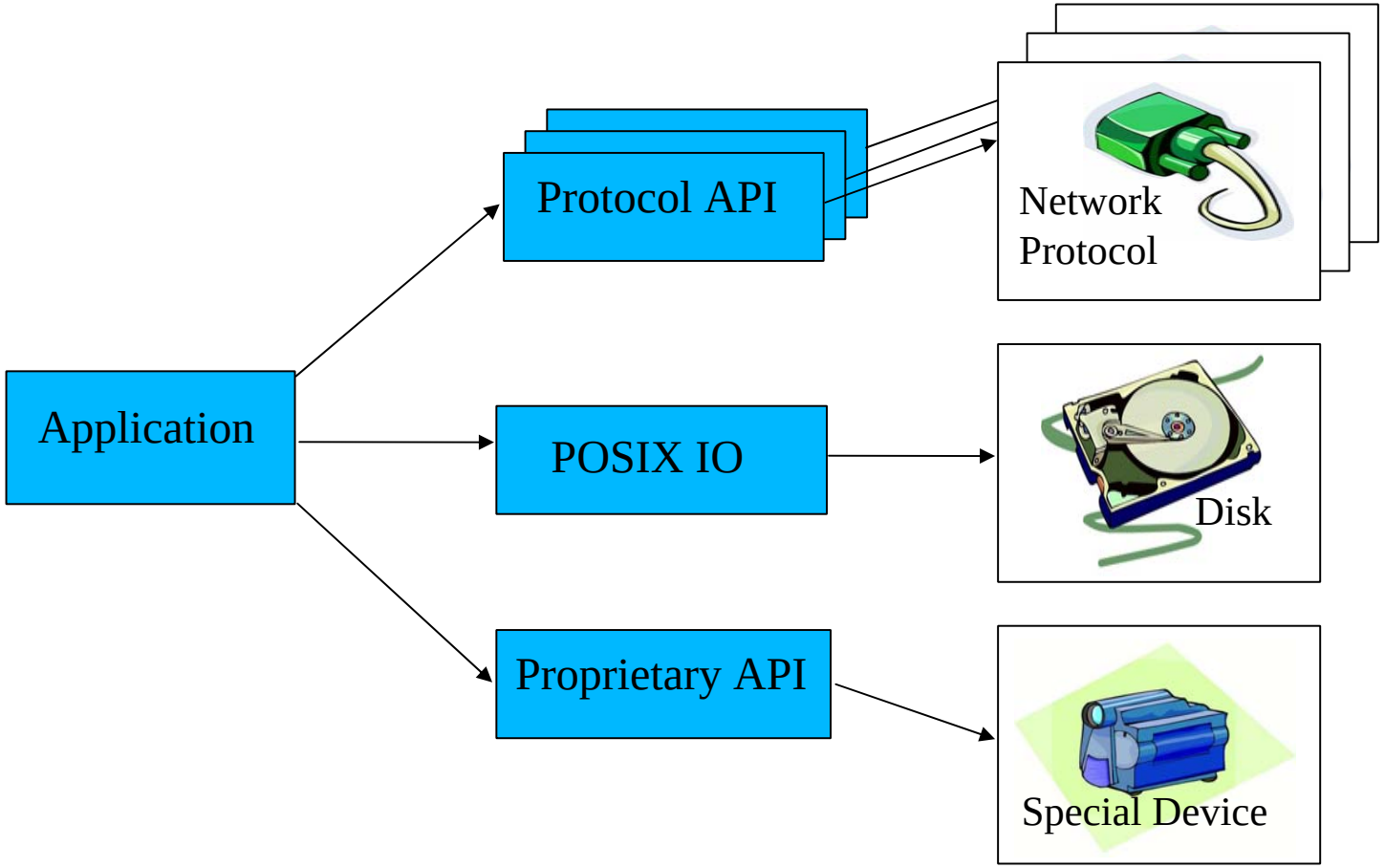
the globus alliance
www.globus.org

Security Alternatives

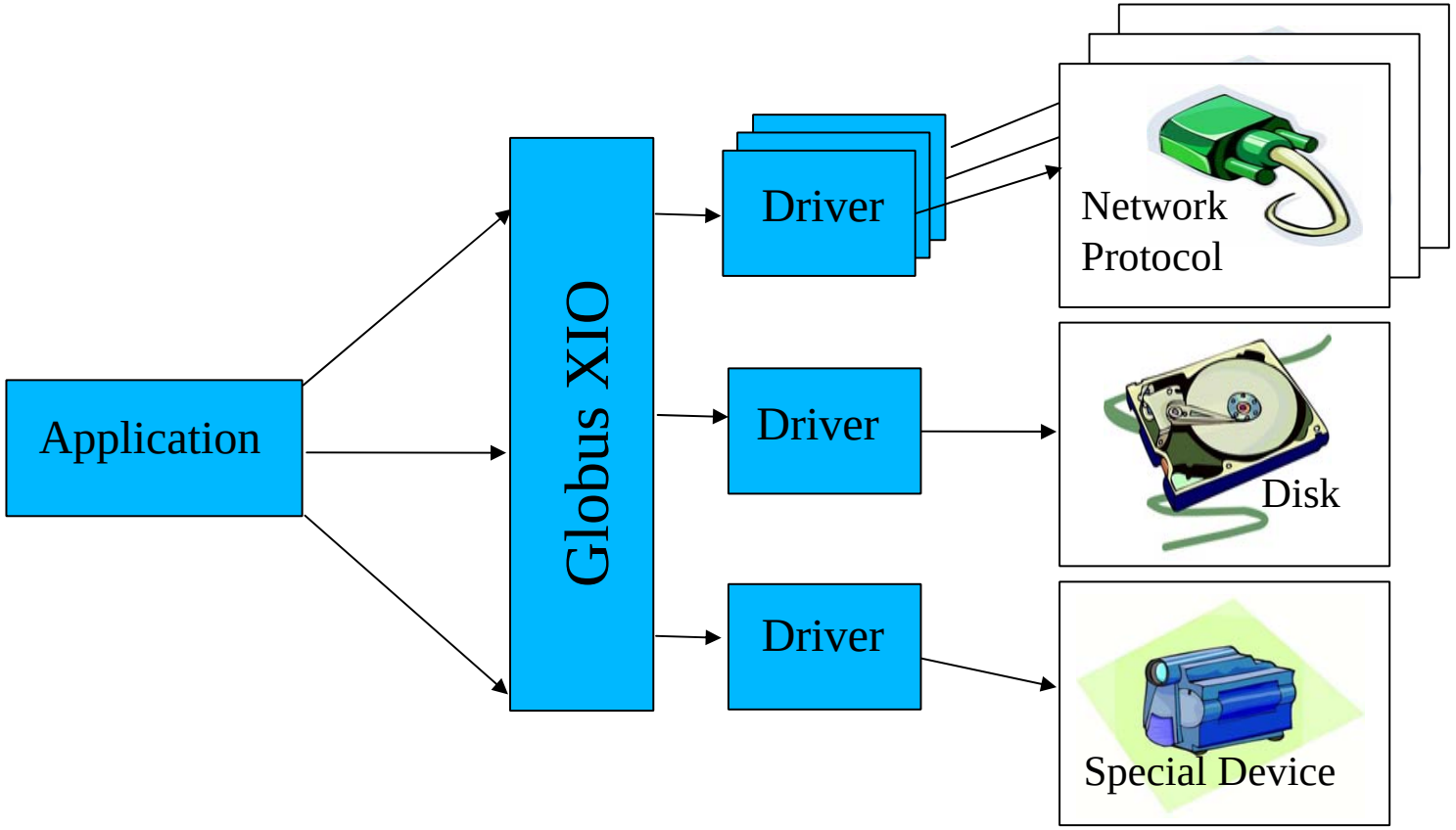
- GSI is very secure, but it is complex and has high startup overhead.
- We added anonymous mode: Anyone can run a transfer if they know the right username
  - -aa –allow-anonymous-name anon
- We added FTP clear text password mode
  - -password-file filename
  - uses the /etc/passwd format, but DON'T use that file or those passwords

Modularity

Typical Application Approach

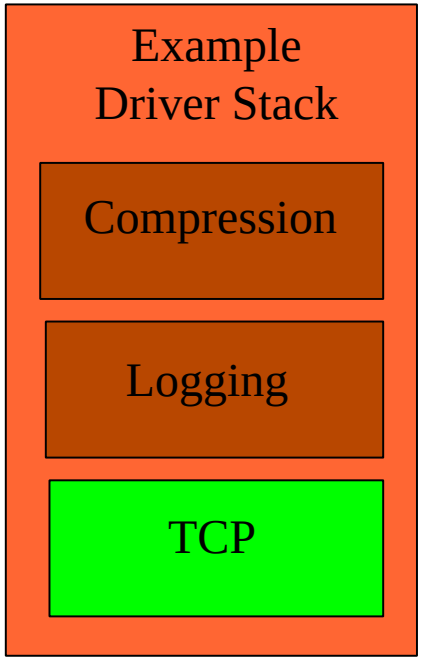


Globus XIO Approach



Stack

- Transport
  - Exactly one per stack
  - Must be on the bottom
- Transform
  - Zero or many per stack
- Control flows from user to the top of the stack, to the transport driver.



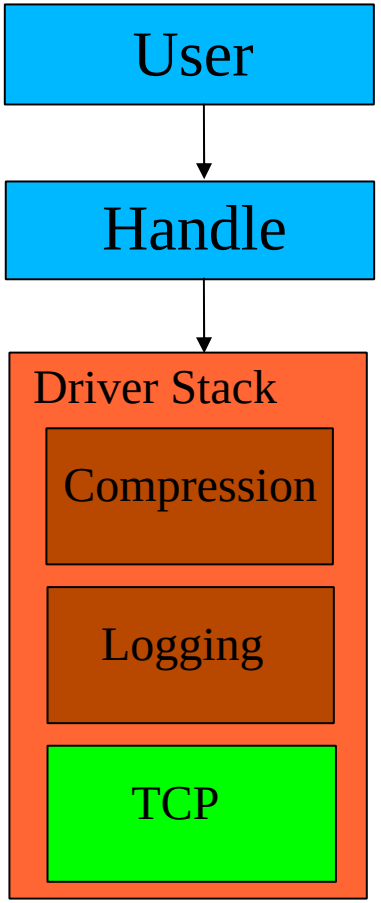
the globus alliance
www.globus.org

Handles

- Associated with a single stack
  - When a handle is opened it is bound to a specific immutable stack.
  - Handle is bound to stack at runtime.
- Used in all user IO operations
  - open/close/read/write
  - Contains the state of the *connection*
- Initialization
  - Driver specific options.

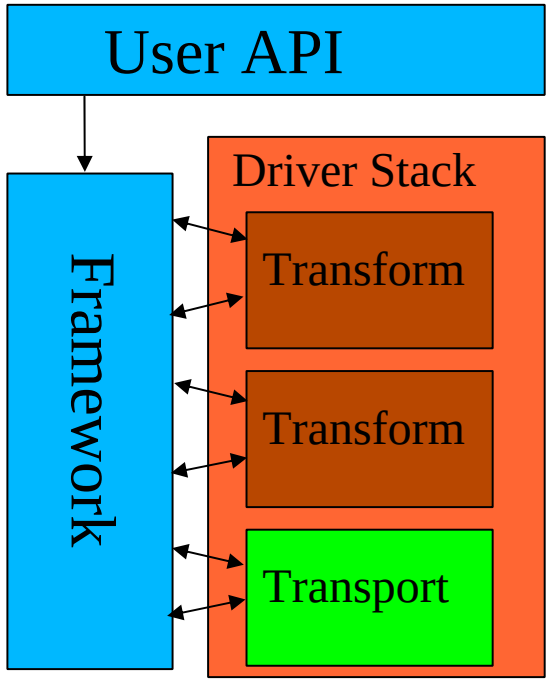
Example Handle Use

- Handle is bound to the stack.
- User performs data operations on the handle.
- The data operation is passed down the stack.
  - The data is compressed by the first driver.
  - The logging driver logs the exchange in syslog.
  - The TCP driver sends the compressed data across the wire.



Globus XIO Framework

- Moves the data from user to driver stack.
- Manages the interactions between drivers.
- Assist in the creation of drivers.
  - Asynchronous support.
  - Close and EOF Barriers.
  - Error checking
  - Internal API for passing operations down the stack.



Driver Development Warning

- For power users only
  - Assumption is you know what you are doing
  - Weaker error reporting
    - For the sake of efficiency.
  - Possible to trip assertions.
    - If the internal API is misused.

Existing Drivers

- Transport
  - TCP, UDP, File
- Transform
  - GSI, HTTP, GSSAPI_FTP
- Advanced
  - UDT, MODE E, GridFTP
- Coming Soon
  - Your driver?

Performance

- XIO was designed to be highly efficient.
- Comparisons of the old globus_io and the old GridFTP server show an improvement of maybe 10-15% (can't guarantee it is statistically significant)
- Had one individual tell us that his protocol implementation actually ran faster when he wrote a driver due to our efficient threading and event handling.

Server Data Storage Interface (DSI)

New Server Architecture

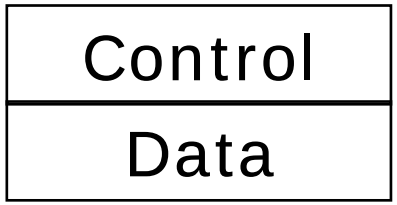
- GridFTP (and normal FTP) use (at least) two separate socket connections:
  - A control channel for carrying the commands and responses
  - A Data Channel for actually moving the data
- Control Channel and Data Channel can be (optionally) completely separate processes.
- A single Control Channel can have multiple data channels behind it.
  - This is how a striped server works.
  - In the future we would like to have a load balancing proxy server work with this.

New Server Architecture

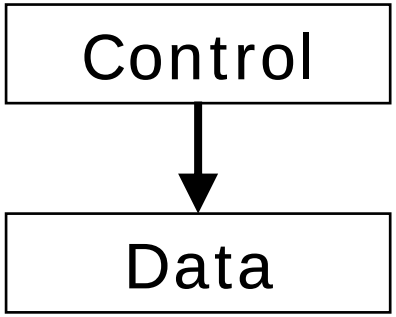
- Data Transport Process (Data Channel) is architecturally, 3 distinct pieces:
  - The protocol handler. This part talks to the network and understands the data channel protocol
  - The Data Storage Interface (DSI). A well defined API that may be re-implemented to access things other than POSIX filesystems
  - ERET/ESTO processing. Ability to manipulate the data prior to transmission.
    - currently handled via the DSI
    - In V4.2 we to support XIO drivers as modules and chaining
- Working with several groups to on custom DSIs
  - LANL / IBM for HPSS
  - UWis / Condor for NeST
  - SDSC for SRB

Possible Configurations

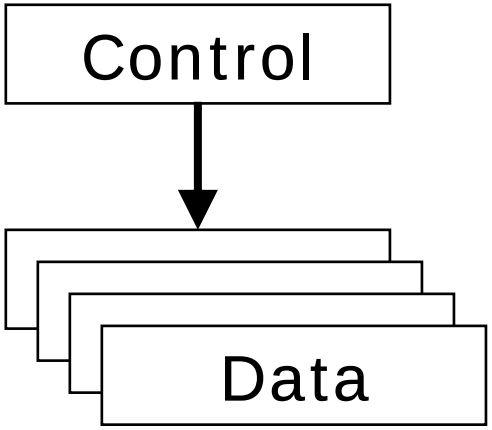
Typical Installation



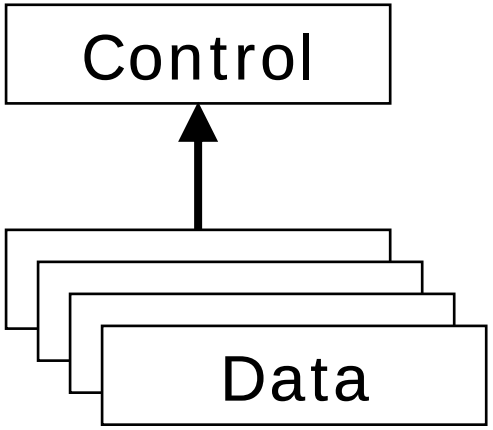
Separate Processes



Striped Server



Striped Server (future)



The Data Storage Interface (DSI)

- Conceptually, the DSI is very simple.
- There are a few required functions (init, destroy)
- Most of the interface is optional, and you can only implement what is needed for your particular application.
- There are a set of API functions provided that allow the DSI to interact with the server itself.
- Note that the DSI could be given significant functionality, such as caching, proxy, backend allocation, etc..

Developer Implemented Functions

- Below is the structure used to hold the pointers to your functions.
- This can be found in <install>/source-trees/gridftp/server/src

```
typedef struct globus_gfs_storage_iface_s
{
    int                                 descriptor;

    /* session initiating functions */
    globus_gfs_storage_init_t           init_func;
    globus_gfs_storage_destroy_t        destroy_func;

    /* transfer functions */
    globus_gfs_storage_transfer_t       list_func;
    globus_gfs_storage_transfer_t       send_func;
    globus_gfs_storage_transfer_t       recv_func;
    globus_gfs_storage_trev_t           trev_func;

    /* data conn funcs */
    globus_gfs_storage_data_t           active_func;
    globus_gfs_storage_data_t           passive_func;
    globus_gfs_storage_data_destroy_t   data_destroy_func;

    globus_gfs_storage_command_t        command_func;
    globus_gfs_storage_stat_t           stat_func;

    globus_gfs_storage_set_cred_t       set_cred_func;
    globus_gfs_storage_buffer_send_t    buffer_send_func;
} globus_gfs_storage_iface_t;
```

the globus alliance
www.globus.org

Master vs. Slave DSI

- If you wish to support process seperation, you will need two DSIs
- The Master DSI will be in the PI or front end. It must implement all functions (that you want to support).
  - Usually, this is relatively trivial and involves minor processing and then “passing” the command over the IPC channel to the slave DSI
- Any functions not implemented will be handled by the server if possible (non-filesystem, active, list)
- All DSI’s must implement the init_func and destroy_func functions.

Slave Functions

- The slave DSI does the real work. It typically implements the following functions:
  - send_func: This function is used to send data from the DSI to the server (get or RETR)
  - recv_func: This function is used to receive data from the server (put or STOR)
  - stat_func: This function performs a unix stat, i.e. it returns file info. Used by the list function
  - command_func: This function handles simple (succeed/fail or single line response) file system operations such as mkdir, site chmod, etc.

Features added

- Memory to Memory Transfers
  - globus-url-copy –p 4 –tcpbs 1000000 ftp://anon@localhost/dev/zero ftp://anon@locahost/dev/null
- Time limit on transfer (in trunk, not 4.0.1)
  - globus-url-copy –t 30 ...
- Others I am forgetting?

Future Plans

- Better doc on XIO driver development
- Better doc on Data Storage Interface development
- Separate data movement from file system ops to ease DSI development
- STILL want to have a development workshop, but need the doc done first ☹
- More extension points (primarily callouts)
- Trivial Testbed setup?
- Your features?