The Globus GridFTP Framework and Server

Raj Kettimuthu

Math & Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, U.S.A.
Outline

- Motivation
- GridFTP Protocol
- Globus GridFTP design and architecture
- Performance
- New features
- Summary
Motivation
Motivation

- Science is increasingly data-driven
- Geographically distributed communities of scientists need to access and analyze large amounts of data
  - Simulation science applications such as climate modeling
  - Experimental science applications such as high-energy physics
- Rapid increase in raw capacity of wide area network - feasible to move large amounts of data across WAN
Motivation

- NSF TeraGrid links large clusters and storage systems at nine sites
  - With a network providing up to 30 Gbit/s
- In principle, move data across this network at > 3 Gbyte/s or 10 Tbyte/hr
- In practice orchestration of such transfers is technically challenging
  - Exploit parallelism in multiple dimension
  - Deal with failures of various sorts
Motivation

- Effective end-to-end data transfers thus demand a systems approach
  - File systems, computers, network interfaces, network protocols - managed in an integrated fashion to meet performance & robustness goals.

- These considerations motivate the work that I describe here
  - Design, implementation & evaluation of a modular & extensible data transfer system suitable for wide area and high-performance environments.
The GridFTP Protocol
What is GridFTP?

- A secure, robust, fast, efficient, standards based, widely accepted data transfer protocol

- A Protocol
  - Multiple Independent implementation can interoperate
    - This works. Fermi Lab has an implementation with their DCache system and U. Virginia has a .Net implementation that work with ours.
    - Lots of people have developed clients independent of the Globus Project.

- The Globus Toolkit supplies a reference implementation:
  - Server
  - Client tools (globus-url-copy)
  - Development Libraries
GridFTP: The Protocol

- 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
Understanding GridFTP

- GridFTP (and normal FTP) use (at least) two separate socket connections
- Control Channel
  - Command/Response
  - Basic file system operations eg. mkdir, delete etc
  - Used to establish data channels
- Data channel
  - Pathway over which file is transferred
  - Many different underlying protocols can be used
    - MODE command determines the protocol
GridFTP (and normal FTP) has 3 distinct components:

- Client and server protocol interpreters which handle control channel protocol
- Data Transfer Process which handles the accessing of actual data and its movement via the data channel
Simple Third Party Transfer

- Client initiates data transfer between 2 servers.
- Client forms CC with 2 servers.
- Commands routed through the client to establish DC between the two servers.
- Data flows directly between servers.
  - Client is notified by each server PI when the transfer is complete.
Control Channel Establishment

- Server listens on a well-known port (2811)
- Client form a TCP Connection to server
- Banner message
- Authentication
  - Anonymous
  - Clear text USER <username>/PASS <pw>
  - Base 64 encoded GSI handshake - Grid Security Infrastructure based on PKI
- Accepted / Rejected
Data Channel Establishment

● PASV command
  ◆ Sent to the *passive* side of the transfer
  ◆ Listen for a connection and reply with the contact information (IP address and port)

● PORT IP,PORT
  ◆ Actively establish a connection to a given passive listener
Data Channel Establishment

Client PI

PASV

PORT <IP:PORT>

Auth

227 <IP:PORT>

Server PI

DTP

Connect

IP:PORT

Server PI

DTP
Data Channel Protocols

- **MODE Command**
  - Allows the client to select the data channel protocol

- **MODE S**
  - Stream mode, no framing
  - Legacy RFC959

- **MODE E**
  - GridFTP extension
  - Parallel TCP streams
  - Data channel caching

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Size</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>(8 bits)</td>
<td>(64 bits)</td>
<td>(64 bits)</td>
</tr>
</tbody>
</table>
What issues are we addressing?

- **Striping**
  - Storage systems are often clusters, and we need to be able to utilize all of that parallelism

- **Collective Operations**
  - Essentially, the striping should be invisible to the outside world

- **Uniform interface**
  - Ideally, any data source can be treated the same way
What issues are we addressing?

- **Network Protocol Independence**
  - TCP has well known issues with high Bandwidth-Delay Product networks
  - Need to be able to take advantage of aggressive protocols on circuits.

- **Diverse Failure Modes**
  - Much happening under the covers, so must be resilient to failures

- **End-to-End Performance**
  - We need to be able to manage performance for a wide range of resources
What did the GridFTP protocol add?

- **Extended Block Mode**
  - Data is sent in “packets” with a header containing a 64 bit offset and length
  - Allows out-of-order reception of packets

- **Restart and Performance Markers**
  - Allows for robust restart and perf monitoring

- **SPAS/SPOR**
  - Striped PASV and striped PORT
  - Allows a list of IP/ports to be returned
What did the GridFTP Protocol add?

- **Data Channel Authentication**
  - Needed since in third party transfer, you don’t know who will connect to the listener.

- **ESTO/ERET**
  - Allows for additional processing on the data prior to storage/transmission
  - We use this for partial file transfers

- **SBUF/ABUF**
  - Manual and automatic TCP buffer tuning

- **Options to set parallelism/striping parameters**
Parallelism vs Striping
Architecture / Design of our Implementation
Description of transfer: completely server-internal communication. Protocol is unspecified and left up to the implementation.

Info on transfer: restart markers, performance markers, etc. Server PI optionally processes these, then sends them to the client PI.
Possible Configurations

Typical Installation

Control
Data

Separate Processes

Control
Data

Striped Server

Control

Data

Striped Server (future)

Control

Data
Data Transfer Processor
Data Storage Interface

- This is a very powerful abstraction
- Several can be available and loaded dynamically via the ERET/ESTO commands
- Anything that can implement the interface can be accessed via the GridFTP protocol
- We have implemented
  - POSIX file (used for performance testing)
  - HPSS (tape system; IBM)
  - Storage Resource Broker (SRB; SDSC)
  - NeST (disk space reservation; UWis/Condor)
Globus Extensible IO (XIO) System

- XIO framework presents a standard open/close/read/write interface to many different protocol implementations
  - including TCP, UDP, HTTP -- and now UDT
- The protocol implementations are called drivers.
  - A driver can be dynamically loaded and stacked by any Globus XIO application.
Globus XIO Approach

Application → Globus XIO → Driver

Globus XIO

Driver

Network Protocol

Disk

Special Device
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.
Performance Results
Experimental results

● Three settings
  ♦ LAN - 0.2 msec RTT and 622 Mbit/s
  ♦ MAN - 2.2 msec RTT and 1 Gbit/s
  ♦ WAN - 60 msec RTT and 30 Gbit/s
  ♦ MAN - Distributed Optical Testbed in the Chicago area
  ♦ WAN - TeraGrid link between NCSA in Illinois and SDSC in California - each individual has a 1Gbit/s bottleneck link
Experimental results - LAN

![Graph showing bandwidth (Mbit/s) vs. number of streams for different tools: iperf, globus mem, globus disk, and bonnie.]}
Experimental results - MAN

The diagram shows the relationship between the number of streams and the bandwidth (Mbit/s) for different applications:

- **iperf**
- **globus mem**
- **globus disk**
- **bonnie**

The x-axis represents the number of streams, while the y-axis represents the bandwidth in Mbit/s.
Experimental results - WAN

The graph shows the bandwidth (Mbit/s) measured by various tools as a function of the number of streams. The tools include:

- iperf
- globus mem
- globus disk
- bonnie

The y-axis represents bandwidth in Mbit/s, while the x-axis represents the number of streams.
Memory to Memory Striping Performance

Bandwidth (Mbit/s) vs. Degree of Striping

- # Stream = 1
- # Stream = 2
- # Stream = 4
- # Stream = 8
- # Stream = 16
- # Stream = 32
Disk to Disk Striping Performance

![Graph showing Disk to Disk Striping Performance](image)

- **Y-axis:** Bandwidth (Mbit/s)
- **X-axis:** Degree of Striping
- **Legend:**
  - # Stream = 1
  - # Stream = 2
  - # Stream = 4
  - # Stream = 8
  - # Stream = 16
  - # Stream = 32

**Notes:**
- The graph illustrates the performance of disk-to-disk striping under varying degrees of striping with different numbers of streams.
- The bandwidth (in Mbit/s) increases as the degree of striping increases.
- The performance improves with an increasing number of streams.

**Source:**
- National Tsing Hua University
- Date: 12/04/07
Scalability tests

- Evaluate performance as a function of the number of clients
- DiPerf test framework to deploy the clients
- Ran server on a 2-processor 1125 MHz x86 machine running Linux 2.6.8.1
  - 1.5 GB memory and 2 GB swap space
  - 1 Gbit/s Ethernet network connection and 1500 B MTU
- Clients created on hosts distributed over PlanetLab and at the University of Chicago (UofC)
Scalability Results

Left axis - load, response time, memory allocated
Right axis - Throughput and CPU %
Scalability results

- 1800 clients mapped in a round robin fashion on 100 PlanetLab hosts and 30 UofC hosts
- A new client created once a second and ran for 2400 seconds
  - During this time, repeatedly requests the transfer of 10 Mbyte file from server’s disk to client’s /dev/null
- Total of 150.7 Gbytes transferred in 15,428 transfers
Scalability results

- Server sustained 1800 concurrent requests with 70% CPU and 0.94 Mbyte memory per request
- CPU usage, throughput, response time remain reasonable even when allocated memory exceeds physical memory
  - Meaning paging is occurring
New features
SSH security mechanism for GridFTP

- GridFTP traditionally uses GSI for establishing secure connections
- In some situations, preferable to use SSH security mechanism
- Leverages the fact that an SSH client can remotely execute programs by forming a secure connection with SSHD
SSHFTP Interactions

- CPI
- GridFTP Server
- USER
- PORT 22
- ROOT
- Stdin/out

Diagram showing interactions between CPI, GridFTP Server, and ROOT/USER.
GridFTP over UDT

- UDT is an application-level data transport protocol that uses UDP to transfer data
- Implement its own reliability and congestion control mechanisms
- Achieves good performance on high-bandwidth, high-delay networks where TCP has significant limitations
- GridFTP uses Globus XIO interface to invoke network I/O operations
  - Created an XIO driver for UDT reference implementation
  - Enabled GridFTP to use it as an alternate transport protocol
## GridFTP over UDT

<table>
<thead>
<tr>
<th></th>
<th>Argonne to NZ Throughput in Mbit/s</th>
<th>Argonne to LA Throughput in Mbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iperf – 1 stream</td>
<td>19.7</td>
<td>74.5</td>
</tr>
<tr>
<td>Iperf – 8 streams</td>
<td>40.3</td>
<td>117.0</td>
</tr>
<tr>
<td>GridFTP mem TCP – 1 stream</td>
<td>16.4</td>
<td>63.8</td>
</tr>
<tr>
<td>GridFTP mem TCP – 8 streams</td>
<td>40.2</td>
<td>112.6</td>
</tr>
<tr>
<td>GridFTP disk TCP – 1 stream</td>
<td>16.3</td>
<td>59.6</td>
</tr>
<tr>
<td>GridFTP disk TCP – 8 streams</td>
<td>37.4</td>
<td>102.4</td>
</tr>
<tr>
<td>GridFTP mem UDT</td>
<td>179.3</td>
<td>396.6</td>
</tr>
<tr>
<td>GridFTP disk UDT</td>
<td>178.6</td>
<td>428.3</td>
</tr>
<tr>
<td>UDT mem</td>
<td>201.6</td>
<td>432.5</td>
</tr>
<tr>
<td>UDT disk</td>
<td>162.5</td>
<td>230.0</td>
</tr>
</tbody>
</table>
Summary

- The GridFTP protocol provides a good set of features for data movement requirements in the Grid.
- The Globus implementation of this protocol provides a flexible design / architecture for integrating with different communities, storage systems, and protocols.
- Our implementation is robust and performant over a range of environments.
Questions?