



GT4 GridFTP for Users: The New GridFTP Server

Bill Allcock, ANL
NeSC, Edinburgh, Scotland
Jan 27-28, 2005



Outline

- Quick Class Survey
- Basic Definitions
- GridFTP Overview
- globus-url-copy
 - ◆ URL syntax
 - ◆ command line options
 - ◆ exercise: Lets move some files
 - ◆ exercise: using debug with globus-url-copy
- Other clients
 - ◆ RFT client
 - ◆ UberFTP



Outline

- Troubleshooting
 - ◆ no proxy
 - ◆ CA not trusted
 - ◆ Firewall problems
 - ◆ bad source file
- Running a server as a user
 - ◆ personal mode
 - ◆ Simple CA
- GridFTP, TCP, and the Bandwidth Delay Product (BWDP)
 - ◆ Exercise: Calculating the BWDP
 - ◆ Exercise: Checking TCP configuration of your machine.
 - ◆ iperf



the globus alliance

www.globus.org

Running the Server as a User the Prelude

- In a shell, do the following:
 - ◆ `cd ~`
 - ◆ `wget`
 - ◆ `gunzip .tar.gz`
 - ◆ `tar -xvf .tar`
 - ◆ `cd gt`
 - ◆ `configure --prefix=<your home>/gridftp --flavor=gcc32dbg`
 - ◆ `make prewsgridftp postinstall`
 - ◆ You just built GridFTP



the globus alliance

www.globus.org

Quick Class Survey

- By show of hands, how many...
 - ◆ Know what GridFTP is?
 - ◆ Can describe the difference between a client and a server (for GridFTP)?
 - ◆ Know the difference between a control channel and a data channel?
 - ◆ Have used globus-url-copy before?
 - ◆ install their own software on Linux?
 - ◆ Know what a bandwidth delay product is?

Basic Definitions



Basic Definitions

- **Command – Response Protocol**
 - ◆ A client can only send one command and then must wait for a “Finished response” before sending another
 - ◆ GridFTP and FTP fall into this category
- **Client**
 - ◆ Sends commands and receives responses
- **Server**
 - ◆ Receives commands and sends responses
 - ◆ Implies it is listening on a port somewhere



Basic Definitions

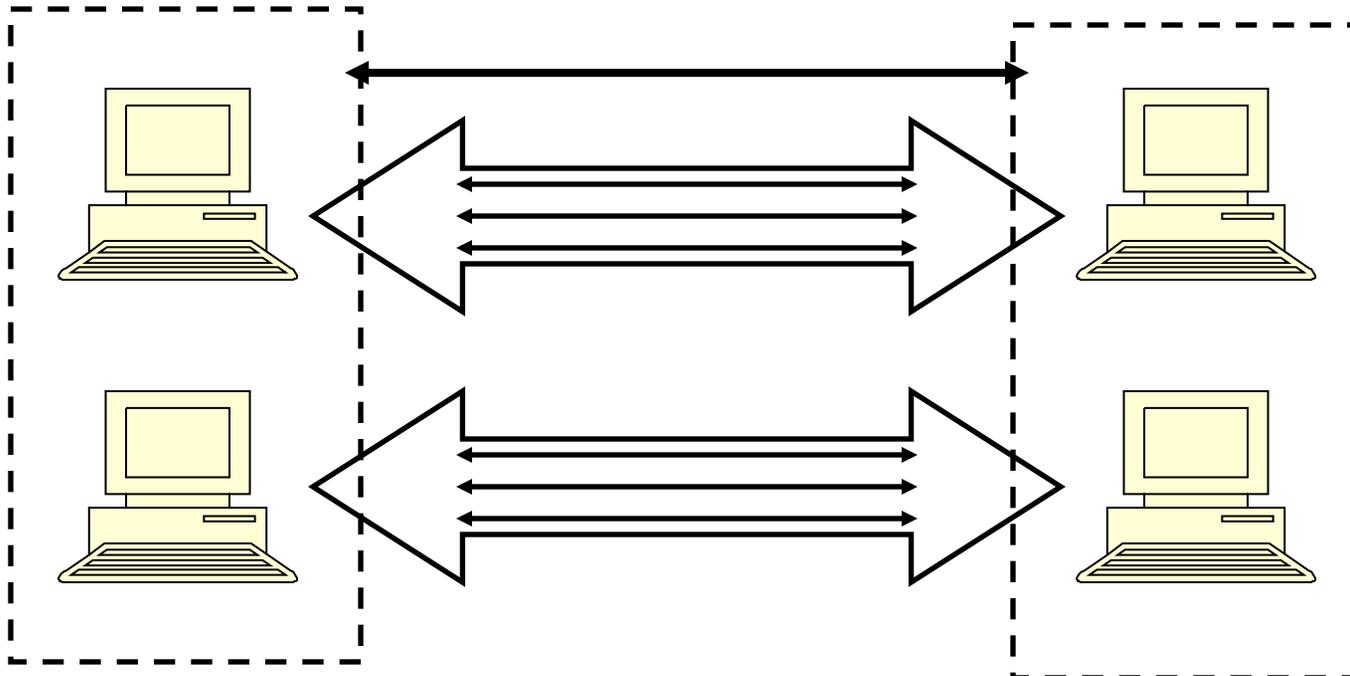
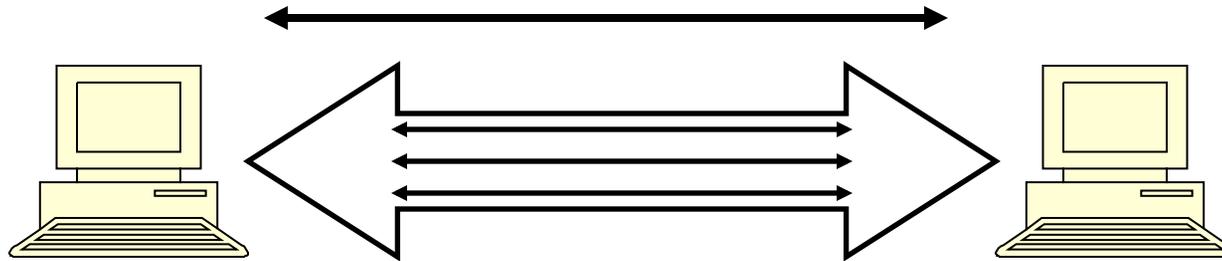
- **Control Channel**
 - ◆ Communication link (TCP) over which commands and responses flow
 - ◆ Low bandwidth; encrypted and integrity protected by default
- **Data Channel**
 - ◆ Communication link(s) over which the actual data of interest flows
 - ◆ High Bandwidth; authenticated by default; encryption and integrity protection optional



Basic Definitions

- **Network Endpoint**
 - ◆ Something that is addressable over the network (i.e. IP:Port). Generally a NIC
 - ◆ multi-homed hosts
 - ◆ multiple stripes on a single host (testing)
- **Parallelism**
 - ◆ multiple TCP Streams between two network endpoints
- **Striping**
 - ◆ Multiple pairs of network endpoints participating in a single logical transfer (i.e. only one control channel connection)

Parallelism vs Striping



GridFTP Overview



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.
- We also supply a reference implementation:
 - ◆ Server
 - ◆ Client tools (globus-url-copy)
 - ◆ Development Libraries



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>



wuftp 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



New GT4 GridFTP Implementation

- NOT web services based
- NOT based on wuftp
- 100% Globus code. No licensing issues.
- Absolutely no protocol change. New server should work with old servers and custom client code.
- Extremely modular to allow integration with a variety of data sources (files, mass stores, etc.)
- Striping support is present.
- Has IPV6 support included (EPRT, EPSV), but we have limited environment for testing.
- Based on XIO
- wuftp specific functionality, such as virtual domains, will NOT be present



the globus alliance

www.globus.org

Extensible IO (XIO) system

- Provides a framework that implements a Read/Write/Open/Close Abstraction
- Drivers are written that implement the functionality (file, TCP, UDP, GSI, etc.)
- Different functionality is achieved by building protocol stacks
- GridFTP drivers will allow 3rd party applications to easily access files stored under a GridFTP server
- Other drivers could be written to allow access to other data stores.
- Changing drivers requires minimal change to the application code.



New Server Architecture

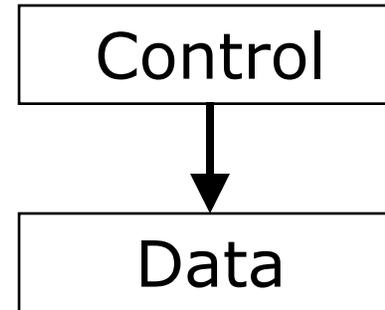
- GridFTP (and normal FTP) use (at least) two separate socket connections:
 - ◆ A control channel for carrying the commands and responses
 - ◆ A Data Channel for actually moving the data
- Control Channel and Data Channel can be (optionally) completely separate processes.
- A single Control Channel can 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.

Possible Configurations

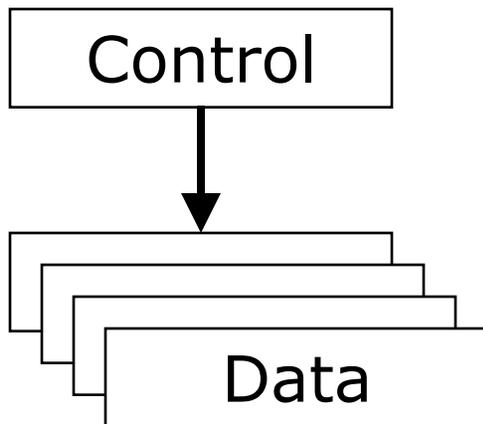
Typical Installation



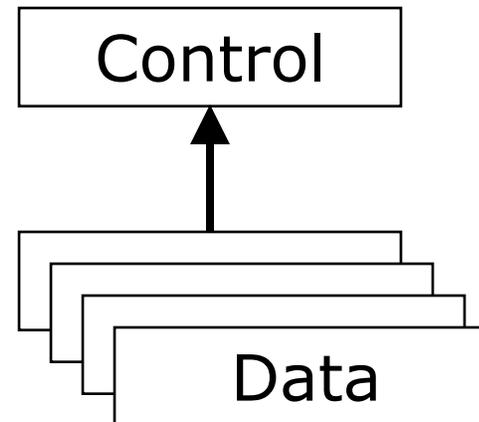
Separate Processes



Striped Server



Striped Server (future)





the globus alliance

www.globus.org

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



The Data Storage Interface (DSI)

- Unoriginally enough, it provides an interface to data storage systems.
- Typically, this data storage system is a file system accessible via the standard POSIX API, and we provide a driver for that purpose.
- However, there are many other storage systems that it might be useful to access data from, for instance HPSS, SRB, a database, non-standard file systems, etc..



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..



the globus alliance

www.globus.org

Current Development Status

- GT3.9.4 has a very solid alpha. This code base has been in use for over a year.
- The data channel code, which was the code we added to wuftp, was re-used and so has been running for several years.
- Initial bandwidth testing is outstanding.
- Stability testing shows non-striped is rock solid
- Striped has a memory leak that we are hunting
- <http://dc-master.isi.edu/mrtg/ned.html>



Status continued

- Stability tests to date have been for a single long running transfer
- We are working on sustained load and “job storm” tests
- A usable response in the face of overload is a key goal.
- Completed an external security architecture review
 - ◆ Likely to make changes to the “recommended configuration”
 - ◆ This is a deployment issue, not a code issue.
- Planning an external code review.



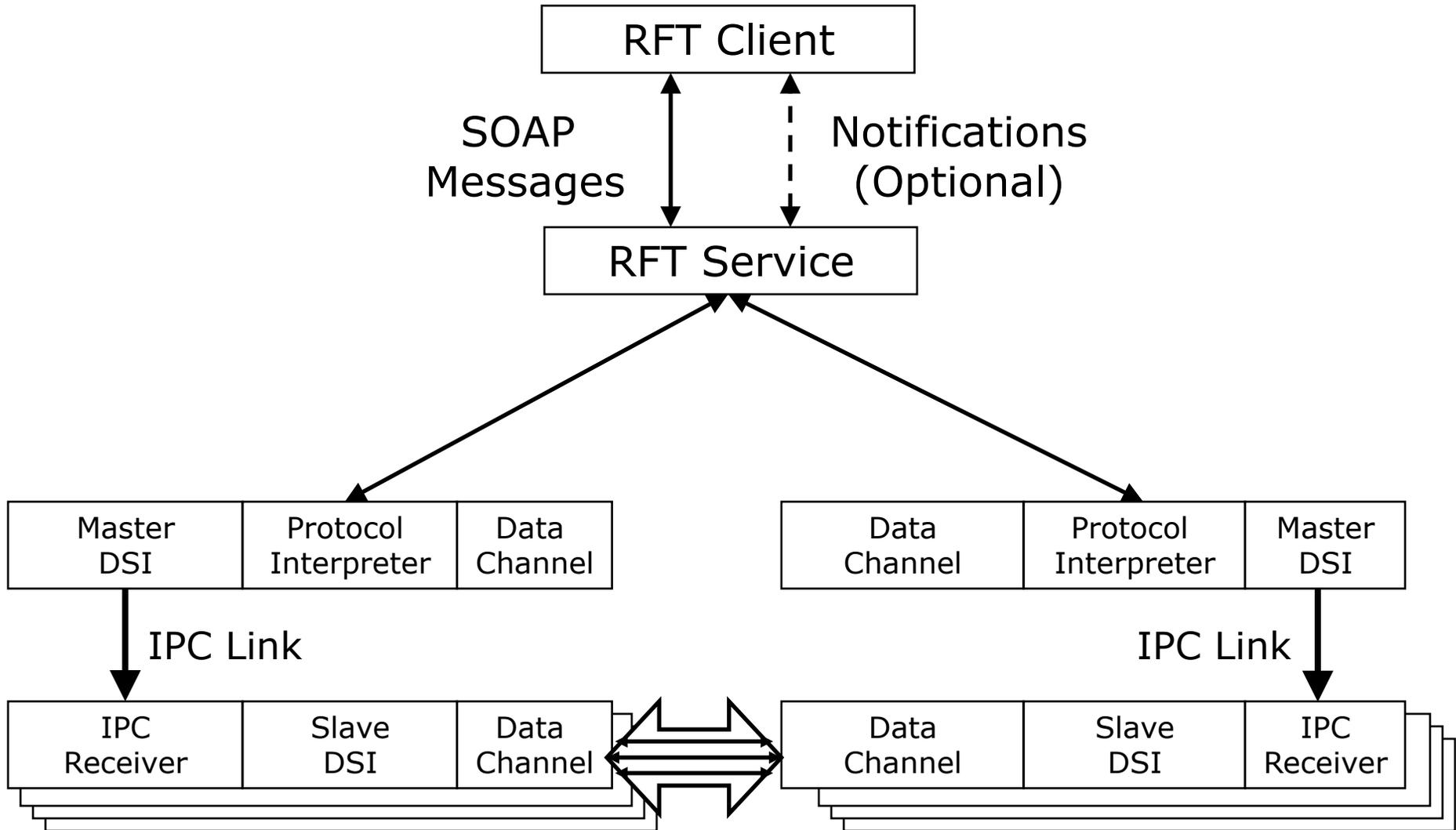
the globus alliance

www.globus.org

Deployment Scenario under Consideration

- All deployments are striped, i.e. separate processed for control and data channel.
- Control channel runs as a user who can only read and execute executable, config, etc. It can write delegated credentials.
- Data channel is a root setuid process
 - ◆ Outside user never connects to it.
 - ◆ If anything other than a valid authentication occurs it drops the connection
 - ◆ It can be locked down to only accept connections from the control channel machine IP
 - ◆ First action after successful authentication is setuid

Third Party Transfer





Striped Server

- Multiple nodes work together and act as a single GridFTP server
- An underlying parallel file system allows all nodes to see the same file system and must deliver good performance (usually the limiting factor in transfer speed)
 - ◆ I.e., NFS does not cut it
- Each node then moves (reads or writes) only the pieces of the file that it is responsible for.
- This allows multiple levels of parallelism, CPU, bus, NIC, disk, etc.
 - ◆ Critical if you want to achieve better than 1 Gbs without breaking the bank

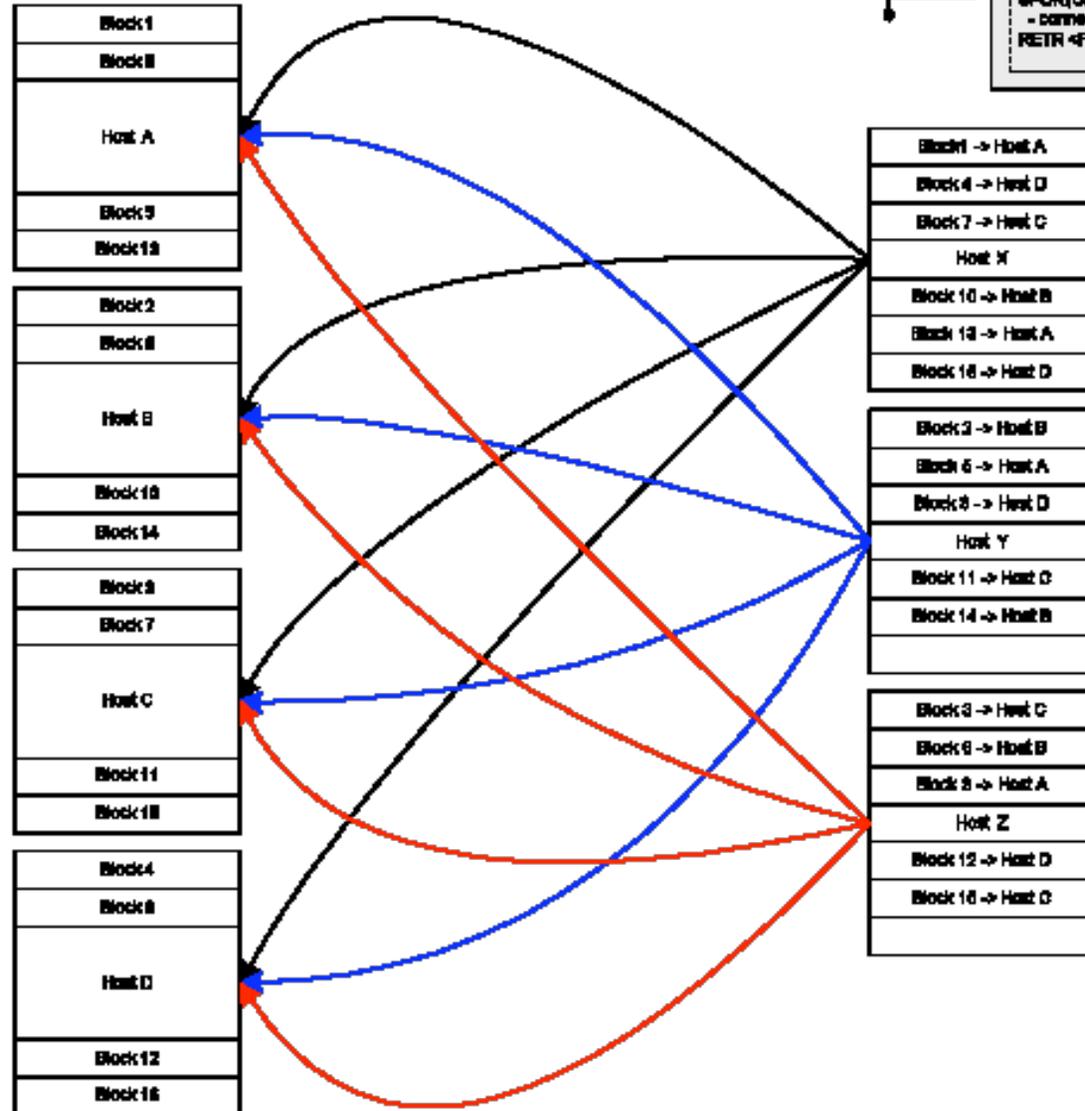


16-Nov-03

GridFTP Striped Transfer

MODE P
SPMS (Listen)
- returns list of host:port pairs
BTOR <File Name>

MODE P
SPOR (Connect)
- connect to the host:port pairs
RETR <File Name>



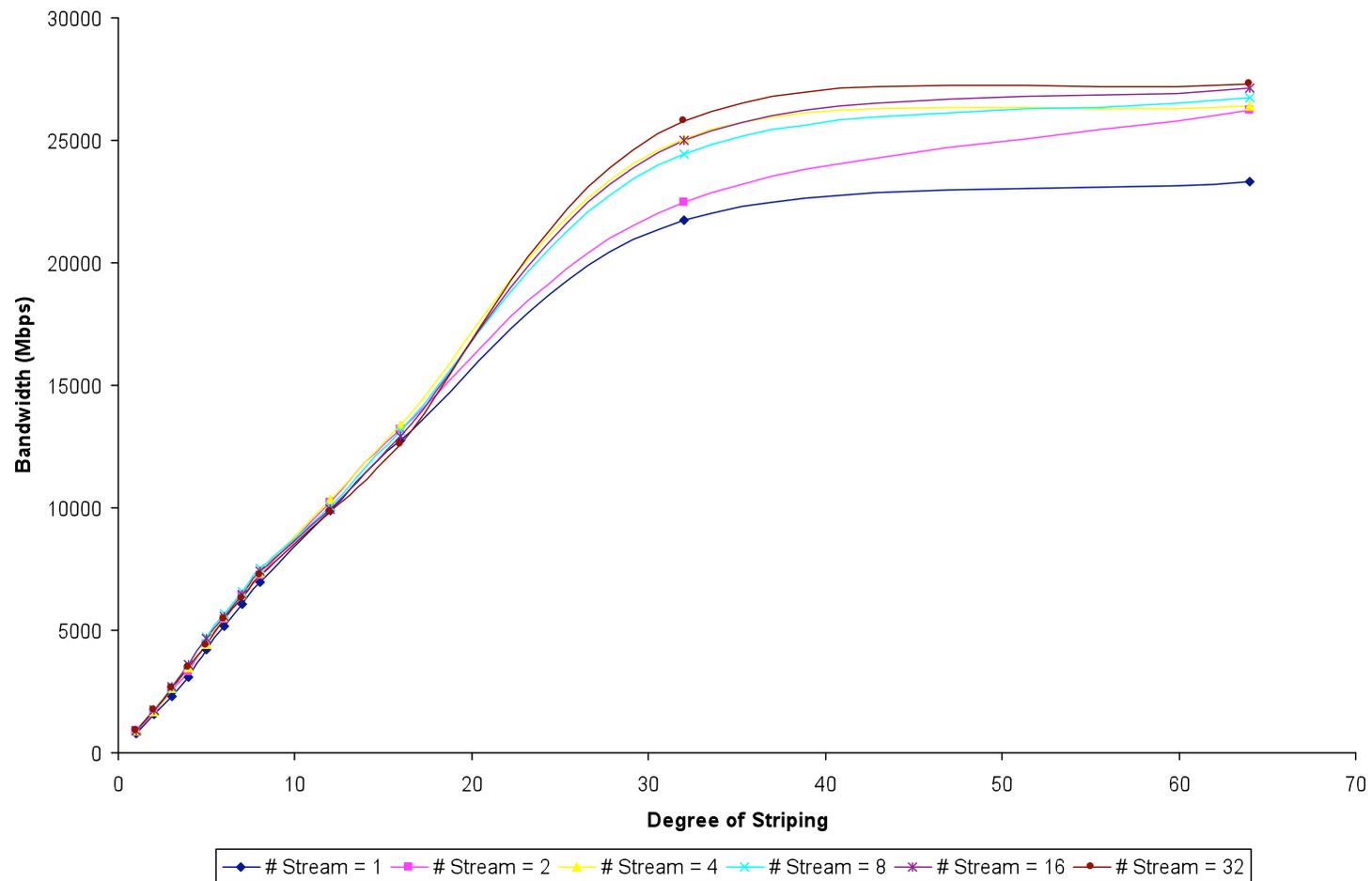


TeraGrid Striping results

- Ran varying number of stripes
- Ran both memory to memory and disk to disk.
- Memory to Memory gave extremely high linear scalability (slope near 1).
- We achieved 27 Gbs on a 30 Gbs link (90% utilization) with 32 nodes.
- Disk to disk we were limited by the storage system, but still achieved 17.5 Gbs

Memory to Memory Striping Performance

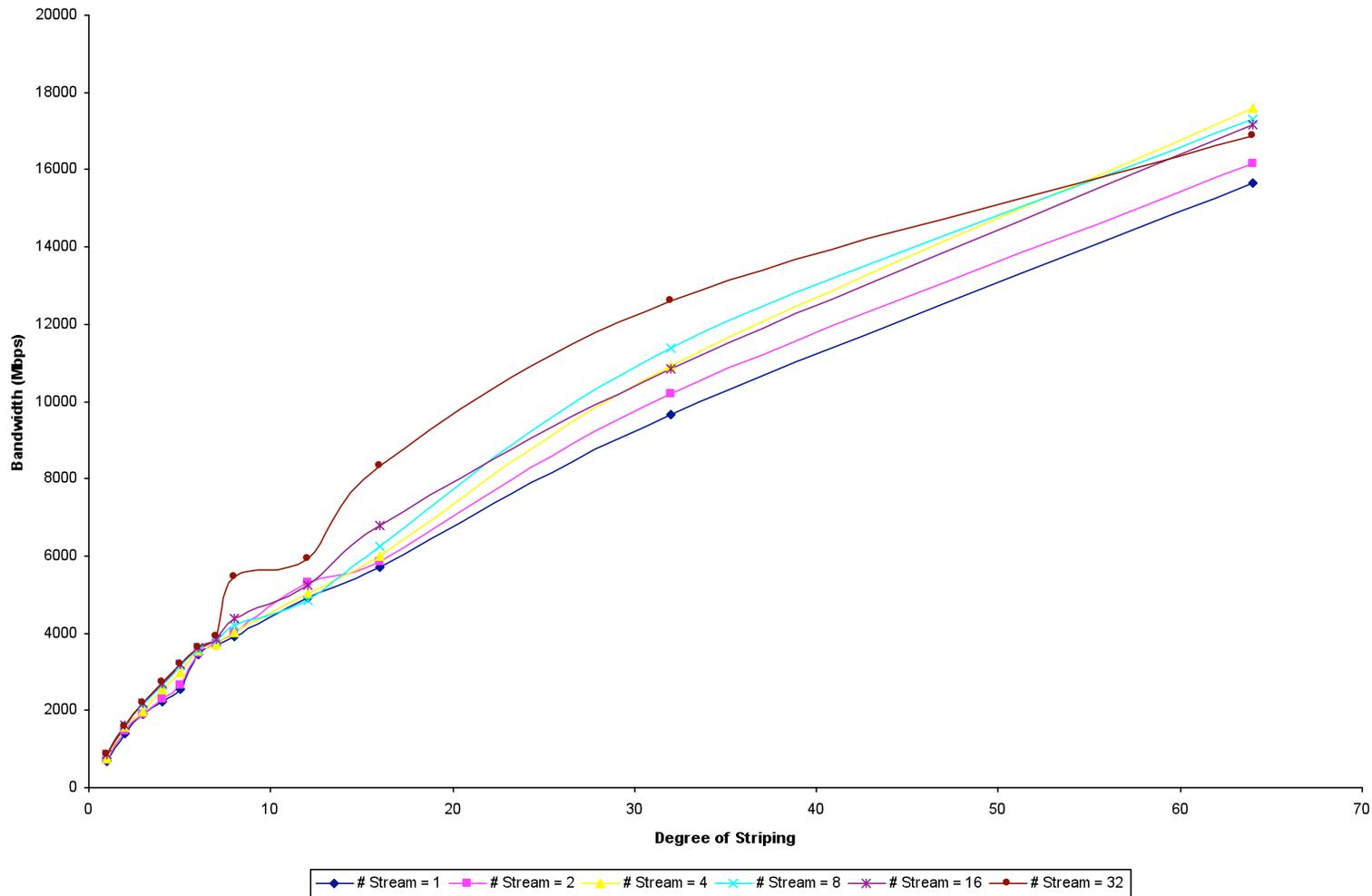
BANDWIDTH Vs STRIPING





Disk to Disk Striping Performance

BANDWIDTH Vs STRIPING





GridFTP: Caveats

- Protocol requires that the sending side do the TCP connect (possible Firewall issues)
- Client / Server
 - ◆ Currently, no simple encapsulation of the server side functionality (need to know protocol), therefore Peer to Peer type apps VERY difficult
 - A library with this encapsulation is on our radar, but no timeframe.
 - ◆ Generally needs a pre-installed server
 - Looking at a “dynamically installable” server

globus-url-copy



Overview

- Command line scriptable client
- Globus does not provide an interactive client
- Most commonly used for GridFTP, however, it supports many protocols
 - ◆ gsiftp:// (GridFTP, historical reasons)
 - ◆ ftp://
 - ◆ http://
 - ◆ https://
 - ◆ file://



Syntax Overview

- `globus-url-copy [options] srcURL dstURL`
- `guc gsiftp://localhost/foo file:///bar`
- `guc -vb -dbg -tcp-bs 1048576 -p 8
gsiftp://localhost/foo gsiftp://localhost/bar`
- `guc https://host.domain.edu/foo ftp://
host.domain.gov/bar`



URL Rules

- protocol://[user:pass@[host]/path
- For guc supported protocols are:
 - ◆ gsiftp:, ftp:, file:, http:, https:
- user:pass is for FTP
 - ◆ GridFTP only accepts that if anonymous login is enabled
- host can be anything resolvable
 - ◆ IP address, localhost, DNS name



URL Rules: Paths

- protocol://[user:pass@][host][:port]/path
- Note that the / between host and path is ***NOT*** part of the path.
- RFC 1738 says paths should be relative to your login directory
 - ◆ Most implementation use root rooted paths
 - ◆ This is the GridFTP default
 - ◆ We support RFC 1738 with a switch
 - ◆ To be root rooted with RFC1738 you start the path with %2F



URL Rules: Paths

- `gsiftp://localhost/tmp/foo`
 - ◆ to you it *looks* like the path is `/tmp/foo`
 - ◆ it really is interpreted as:
 - CD to default directory
 - CD tmp
 - access file foo
 - ◆ so, if the default directory is root you end up accessing `/tmp/foo`
 - ◆ but, if the default directory is your home directory (RFC1738) you end up accessing `~/tmp/foo`
 - ◆ to access `/tmp/foo` with RFC 1738 it would be `gsiftp://localhost/%2F/tmp/foo`



The Options: The Overview

- If you remember nothing else remember this slide
- -p (parallelism or number of streams)
 - ◆ rule of thumb 4-8, start with 4
- -tcp-bs (TCP buffer size)
 - ◆ use either ping or traceroute to determine the RTT between hosts
 - ◆ $\text{buffer size} = \text{BW (Mbs)} * \text{RTT (ms)} * 1000 / 8 / \langle \text{value you used for } -p \rangle$
- -vb if you want performance feedback
- -dbg if you have trouble



The Options: The Details

- `guc -help` gives a good overview
- We are going to look at the web doc



Exercise: Simple File Movement

- `grid-proxy-init`
- `echo test > /tmp/test`
- `guc gsiftp://localhost/tmp/test file:///tmp/test2`
 - ◆ get (from server to client)
- `guc file:///tmp/test2 gsiftp://localhost/tmp/test3`
 - ◆ put (from client to server)
- `guc gsiftp://localhost/tmp/test3`
`gsiftp://<host-next-to-you>/tmp/test4`
 - ◆ Third party transfer (between two servers)



Exercise: Using -dbg

- grid-proxy-destroy
- `guc -dbg -vb gsiftp://localhost/dev/zero
gsiftp://localhost/dev/null`
- grid-proxy-init
- re-run the above
- DEMONSTRATION:
 - ◆ TCP buffer size and streams really do make a difference
 - ◆ Wide area transfer with buffers too small
 - ◆ many streams with buffer too small
 - ◆ done right (see The Options: Overview)



Exercise: Free Time to experiment

- Try different commands and options
- If you have access to other hosts and want to move files there, feel free.



Troubleshooting

- no proxy
 - ◆ grid-proxy-destroy
 - ◆ guc gsiftp://localhost/dev/zero file:///dev/null
 - ◆ add -dbg
 - ◆ grid-proxy-init
 - ◆ guc gsiftp://localhost/dev/zero file:///dev/null
 - ◆ add -dbg



Troubleshooting

- CA not trusted (demonstration)
 - ◆ grid-proxy-destroy
 - ◆ set X509_USER_CERT to my DOE Cert
 - ◆ grid-proxy-init
 - ◆ guc gsiftp://localhost/dev/zero file:///dev/zero
 - ◆ add DOE cert and signing policy to /etc/grid-security (you need root for this)
 - ◆ guc gsiftp://localhost/dev/zero file:///dev/zero



Troubleshooting

- Firewall problems

- ◆ grid-proxy-init

- ◆ `guc gsiftp://localhost:2812/dev/zero file://dev/null`

- Port 2812 is configured to use ports 40000-40100 for data channel and that is blocked by the firewall

- ◆ `guc gsiftp://localhost/dev/zero file:///dev/null`

- port 2811 (the default) is configured to use ports 50000-50100 for the data channel and that is open

- ◆ The only solution is to work with your admins to get a range of ports in the firewall open and the server configured to use it

- remember that for GridFTP the sender MUST connect



Troubleshooting

- **Bad source file**
 - ◆ grid-proxy-init
 - ◆ `guc gsiftp://localhost:2812/tmp/junk file:///tmp/empty`
 - junk does not exist
 - Note that an empty file named empty is created
 - We need to fix this in globus-url-copy, but for now it is there

Running the Server as a User



Check your build

- Hopefully, if built with no problems 😊
- In your terminal window:
 - ◆ `grid-proxy-init`
 - ◆ `<your home>/gridftp/sbin/globus-gridftp-server -p 60000`
 - ◆ `grid-cert-info -subject > ~/.globus/grid-mapfile`
 - ◆ `echo <space> student >> grid-mapfile`
 - ◆ use `globus-url-copy` as usual, but add:
 - `-s `grid-proxy-info -subject``



For extra credit...

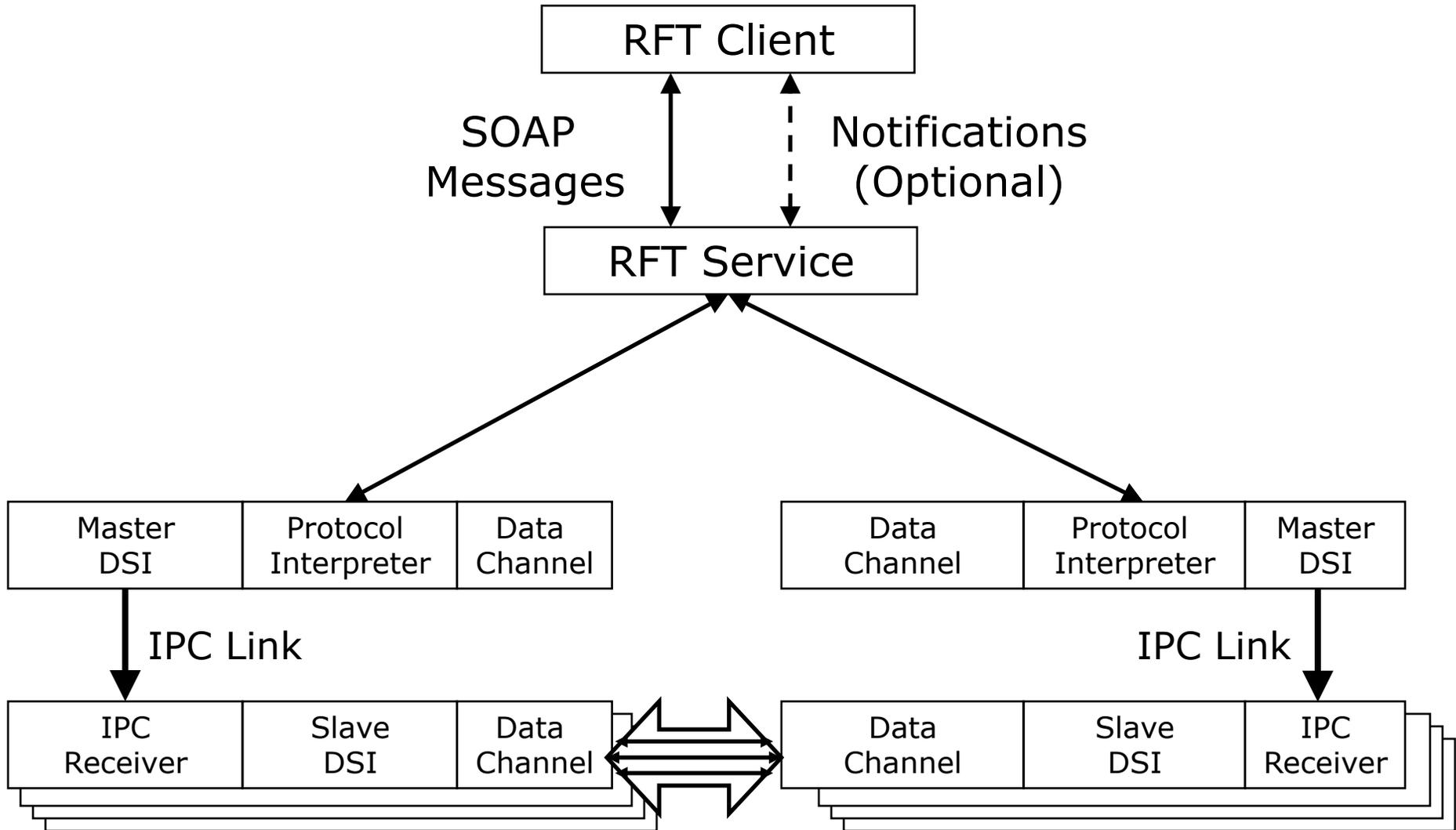
- Add your neighbors subject name to your local grid-mapfile, but map him to your local account
 - ◆ NOTE: In most real life situations, this is a definite NO-NO. You are essentially letting him use your account, which
- Now take turns running 3rd party transfers
 - ◆ You will now have to specify the `-ss` and `-ds` seperately since one server will be running under your proxy and one will be under your neighbors



Other Clients

- Globus also provides a Reliable File Transfer (RFT) service
- Think of it as a job scheduler for data movement jobs.
- The client is very simple. You create a file with source-destination URL pairs and options you want, and pass it in with the `-f` option.
- You can “fire and forget” or monitor its progress.

Third Party Transfer





Other Clients

- Interactive client called UberFTP
- This is NOT from Globus
- It was produced at NCSA for the TeraGrid project
- This is not an endorsement, *we* wont answer bugs, I have never used it, but there are people who use it and like it.

Bandwidth Delay Product



What's wrong with TCP?

- You probably wouldn't be here if you didn't know that.
- TCP was designed for Telnet / Web like applications.
- It was designed when T1 was a fast network, big memory was 2MB, not 2 GB, and a big file transfer was 100MB, not 100GB or even Terabytes.



AIMD and BWDP

- The primary problems are:
 - ◆ Additive Increase Multiplicative Decrease (AIMD) congestion control algorithm of TCP
 - ◆ Requirement of having a buffer equal to the Bandwidth Delay Product (BWDP)
 - ◆ The interaction between those two.
 - ◆ We use parallel and striped transfers to work around these problems.



AIMD

- To the first order this algorithm:
 - ◆ Exponentially increases the congestion window (CWND) until it gets a congestion event
 - ◆ Cuts the CWND in half
 - ◆ Linearly increases the CWND until it reaches a congestion event.
 - ◆ This assumes that congestion is the limiting factor
 - ◆ Note that CWND size is equivalent to Max BW



BWDP

- TCP is reliable, so it has to hold a copy of what it sends until it is acknowledged.
- Use a pipe as an analogy
- I can keep putting water in until it is full.
- Then, I can only put in one gallon for each gallon removed.
- You can calculate the volume of the tank by taking the cross sectional area times the height
- Think of the BW as the area and the RTT as the length of the network pipe.



Recovery Time

$$\text{Recovery Time} = \frac{\text{Bytes to Recover}}{\text{Rate of Recovery}}$$

$$\text{Bytes to Recover} \propto \frac{1}{2} \text{BW} * \text{RTT} \text{ (BWDP)}$$

$$\text{Rate of Recovery} \propto \frac{\text{MTU}}{\text{RTT}}$$

$$\text{Recovery Time} \propto \frac{\frac{1}{2} \text{BW} * \text{RTT}}{\frac{\text{MTU}}{\text{RTT}}} \Rightarrow \frac{\text{RTT}^2 * \text{BW}}{\text{MTU}}$$



Recovery Time for a Single Congestion Event

- T1 (1.544 Mbs) with 50ms RTT \cong 10 KB
 - ◆ Recovery Time (1500 MTU): 0.16 Sec
- GigE with 50ms RTT \cong 6250 KB
 - ◆ Recovery Time (1500 MTU): 104 Seconds
- GigE to Amsterdam (100ms) \cong 1250 KB
 - ◆ Recovery Time (1500 MTU): 416 Seconds
- GigE to CERN (160ms) \cong 2000 KB
 - ◆ Recovery Time (1500 MTU): 1066 Sec (17.8 min)



How does Parallel TCP Help?

- We are basically cheating.... I mean we are taking advantage of loopholes in the system
- Reduces the severity of a congestion event
- Buffers are divided across streams so faster recovery
- Probably get more than your fair share in the router



Reduced Severity from Congestion Events

- Don't put all your eggs in one basket
- Normal TCP your BW Reduction is 50%
 - ◆ $1000 \text{ Mbs} * 50\% = 500 \text{ Mbs Reduction}$
- In Parallel TCP BW Reduction is:
 - ◆ $\text{Total BW} / N \text{ Streams} * 50\%$
 - ◆ $1000 / 4 * 50\% = 125 \text{ Mbs Reduction}$
- Note we are assuming only one stream receives a congestion event



Faster Recovery from Congestion Events

- Optimum TCP Buffer Size is now $BWDP / (N-1)$ where N is number of Streams
- The division by N-1 is because your maximum bandwidth is still the same, you are just dividing it up. The -1 is to leave room so that other streams can take up BW lost by another stream.
- Since Buffers are reduced in size by a factor of $1/N$ so is the recovery time.
- This can also help work around host limitations. If the maximum buffer size is too small for max bandwidth, you can get multiple smaller buffers.



More than your Fair Share

- This part is inferred, but we have no data with which to back it up.
- Routers apply fair sharing algorithms to the streams being processed.
- Since your logical transfer now has N streams, it is getting N times the service it otherwise normally would.
- I am told there are routers that can detect parallel streams and will maintain your fair share, though I have not run into one yet.



What about Striping?

- Typically used in a cluster with a shared file system, but it can be a multi-homed host
- All the advantages of Parallel TCP
- Also get parallelism of CPUs, Disk subsystems, buses, NICs, etc..
- You can, in certain circumstances, also get parallelism of network paths
- This is a much more complicated implementation and beyond the scope of what we are primarily discussing here.



Nothing comes for free...

- As noted earlier, we are cheating.
- Congestion Control is there for a reason
- Buffer limitations may or may not be there for a reason
- Other Netizens may austracize you.



Congestion Control

- Congestion Control is in place for a reason.
- If every TCP application started using parallel TCP, overall performance would decrease and there would be the risk of congestive network collapse.
- Note that in the face of no congestion parallel streams does not help
- In the face of heavy congestion, it can perform worse.



Buffer Limitations

- More often than not, the system limitations are there because that is way it came out of the box.
- It requires root privilege to change them.
- However, sometimes, they are there because of real resource limitations of the host and you risk crashing the host by over-extending its resources.



Checking the TCP configuration

- Linux handles this via the /proc filesystem
- There are 6 values you need to worry about:
 - ◆ /proc/sys/net/core/rmem_max
 - ◆ /proc/sys/net/core/rmem_default
 - ◆ /proc/sys/net/core/wmem_max
 - ◆ /proc/sys/net/core/wmem_default
 - ◆ /proc/sys/net/ipv4/tcp_rmem
 - ◆ /proc/sys/net/ipv4/tcp_wmem



Checking the TCP configuration

- You can check the values by simply doing:
 - ◆ `cat filename`
- You can change them (with root privilege) by:
 - ◆ `echo 8388608 > /proc/sys/net/rmem_max`
- Note that the `/core` variables have a single value, but the `/ipv4` variables have 3 comma separated values min, default, max
- To make things confusing:
 - ◆ The default value for `ipv4` variables take precedence
 - ◆ The max value for `core` variables take precedence



Cheat enough, but not too much

- If your use of parallel TCP causes too many problems you could find yourself in trouble.
 - ◆ Admins get cranky when you crash their machines
 - ◆ Other users get cranky if you are hurting overall network performance.
- Be a good Netizen



the globus alliance

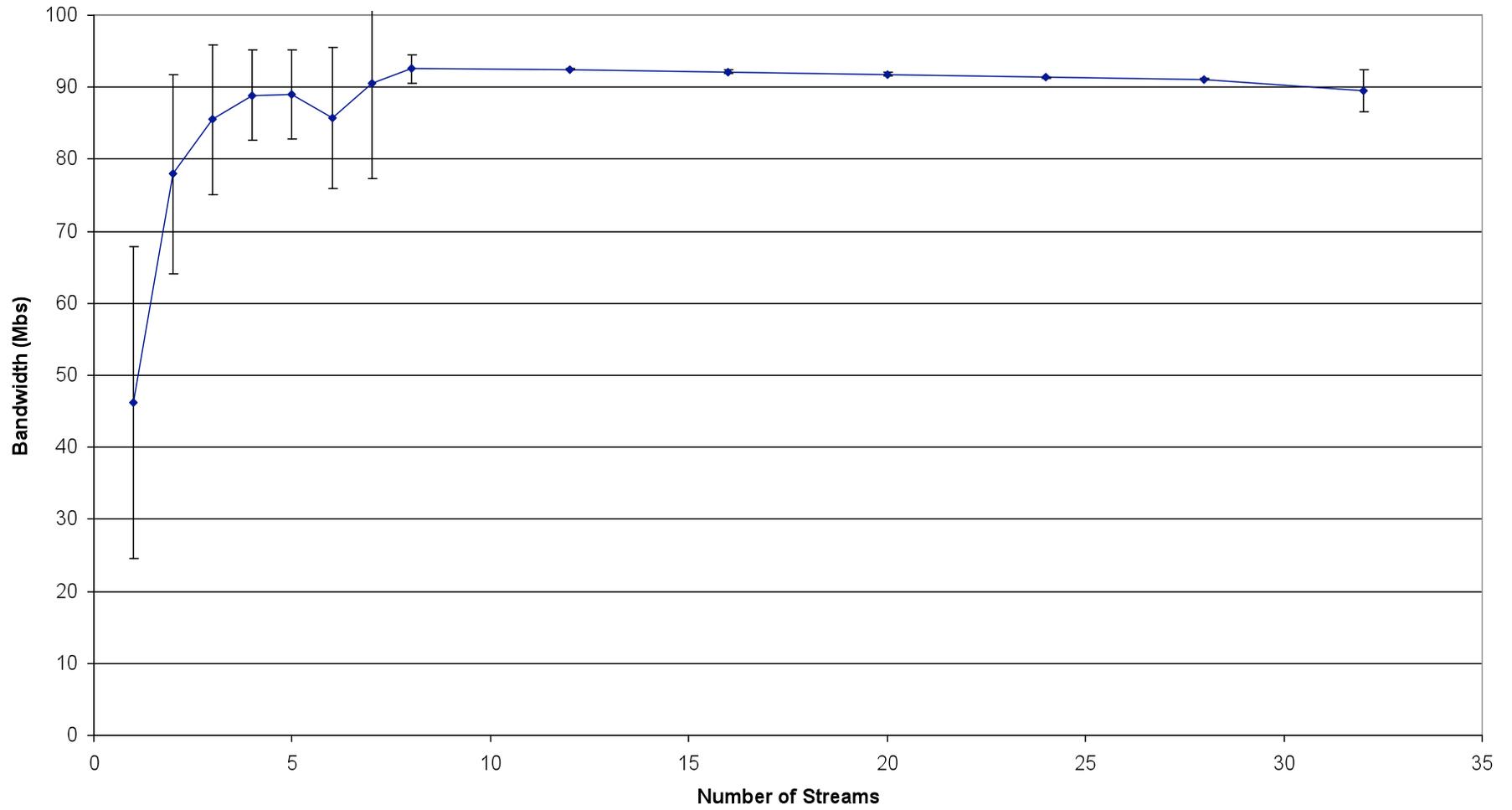
www.globus.org

When should you use Parallel TCP?

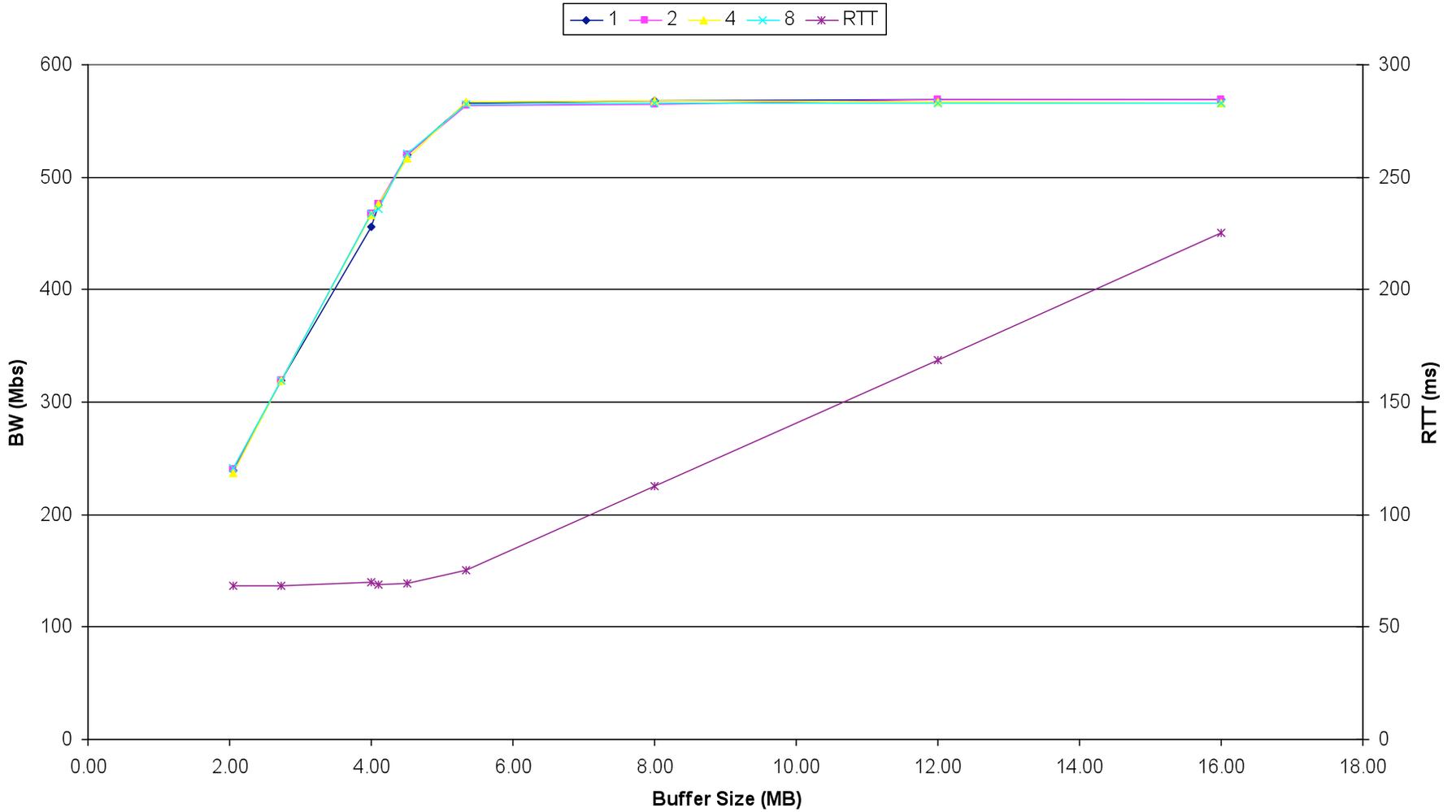
- Engineered, private, semi private, or very over provisioned networks are good places to use parallel TCP.
- Bulk data transport. It makes no sense at all to use parallel TCP for most interactive apps.
- QOS: If you are guaranteed the bandwidth, use it
- Community Agreement: You are given permission to hog the network.
- Lambda Switched Networks: You have your own circuit, go nuts.



Affect of Parallel Streams ANL to ISI



Affect of TCP Buffer Size (iperf)





Exercises

- Calculate the BWDP between here and `arbat.mcs.anl.gov`
- Check the TCP configuration of your machine.
- calculate the BW you should get with 4KB, 8KB, 16KB buffer sizes to `arbat.mcs.anl.gov:6243`
- Demonstration: I will run transfers to compare results



Impact of buffer size

- They can consume substantial resources.