

GridFTP and Cluster Meltdown

When No Means 'Maybe Later'

John Bresnahan

bresnaha@mcs.anl.gov

Argonne National Laboratory

The University of Chicago

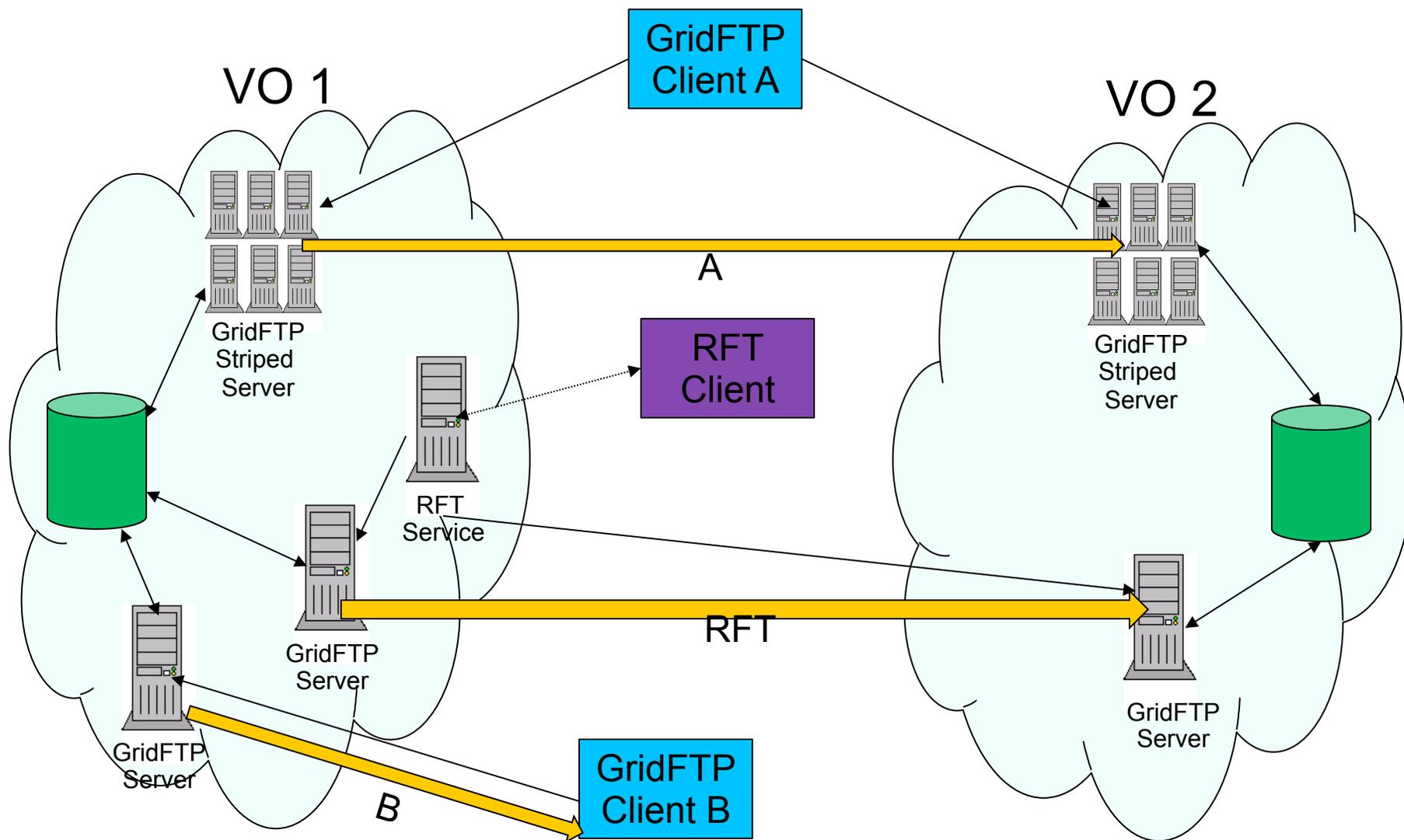


Overview

- Transfer Architectural Components
 - ◆ GridFTP and RFT
 - Subtle intentions of each service
 - How each scales
- Meltdowns
 - ◆ What is one, what isn't one
- Resource protection
 - ◆ Interactions with resources
 - ◆ How to determine limits



Architecture Overview





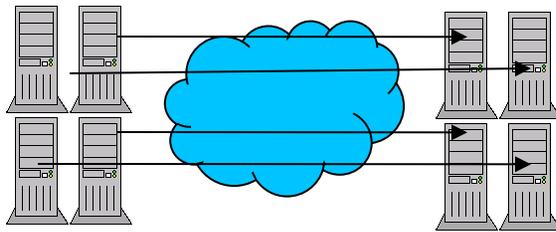
GridFTP Service

- On demand transfer service
 - ◆ When a connection is formed, resources are dedicated
 - ◆ GridFTP might say “not now”
 - ◆ Not a queuing service
- Transfer data as fast as possible
- Maximize resource usage
 - ◆ Without over heating!

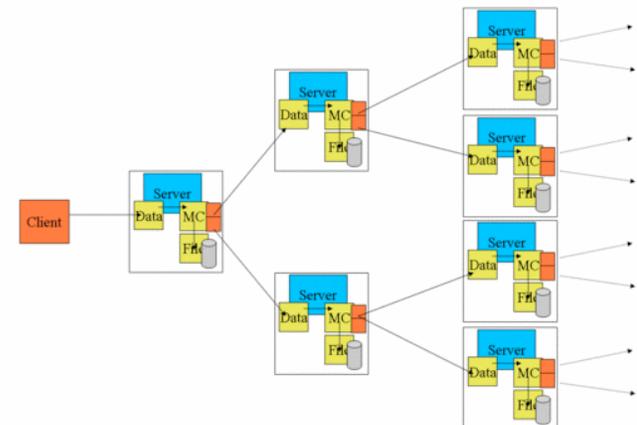


What GridFTP Does

- Fast data transfer service
- Cluster to cluster copy tool



- Intra-cluster broadcast tool
 - ◆ Multi-cast transfers



- Scalable
 - ◆ Need more throughput, add more stripes



RFT Service

- Orchestrates transfers on client's behalf
 - ◆ Third party transfers
 - ◆ Interacts with many GridFTP servers
 - Sees a bigger picture
 - VO level
- Queue requests
 - ◆ RFT should not say no
- Retry requests on failure
 - ◆ Optimizes its workload



What RFT Does

- **Reliable service**
 - ◆ DB backend
 - ◆ Recovers from GridFTP and RFT service failures
- **Batch requests**
 - ◆ Light weight sessions
 - ◆ Submit a Request
 - ◆ Wait for notifications
 - Started, finished, failed, etc



GridFTP: On Demand Service

- Resources are limited
 - ◆ Data transfers are heavy weight operations
 - ◆ Sometimes hardware is too busy
 - Adding another transfer can cause thrashing
 - Collective system throughput goes down
 - ◆ GridFTP might say “no”
- Transfer requests happen immediately
 - ◆ We do not queue, or delay transfers
 - ◆ An established session means an active transfer



the globus alliance

www.globus.org

Why Doesn't GridFTP Queue

- A GridFTP session is heavy weight
 - ◆ Idle sessions consume resources
 - Backward compatible protocol
- Sometimes less is more
 - ◆ Goal: Maximize the collective throughput
 - Sum of all active transfer rates
 - ◆ Too many transfers cause thrashing
 - Results in lower collective throughput
 - ◆ Avoid overheating system resources
- It is in the systems best interest
 - ◆ We know what's good for you 😊



GridFTP Session Resources

- *Even for an idle session*
 - ◆ Active TCP control channel
 - Part of the 959 protocol.
 - A session is defined by a TCP connection
 - ◆ Fork/setuid process
 - Robustness
 - File system/OS permissions
 - ◆ OS buffer space
 - Data channels require large TCP OS buffers
- *Active transfers*
 - ◆ Lots of memory/Net/Disk IO
 - Avoid too small of partitions



If GridFTP Always Said Yes

- OOM: the out of memory handle
 - ◆ OS optimistic provision of TCP buffers
 - ◆ Random processes will be killed
 - ◆ Meltdown
- Shared FS overuse
 - ◆ Pushing the I/O throughput beyond optimal
 - ◆ Causing OOM on IOD machines
- Shares of bandwidth too small
 - ◆ 1 Million transfers at 500b/s each?
 - ◆ OR 10 transfers at 100Mb/s each



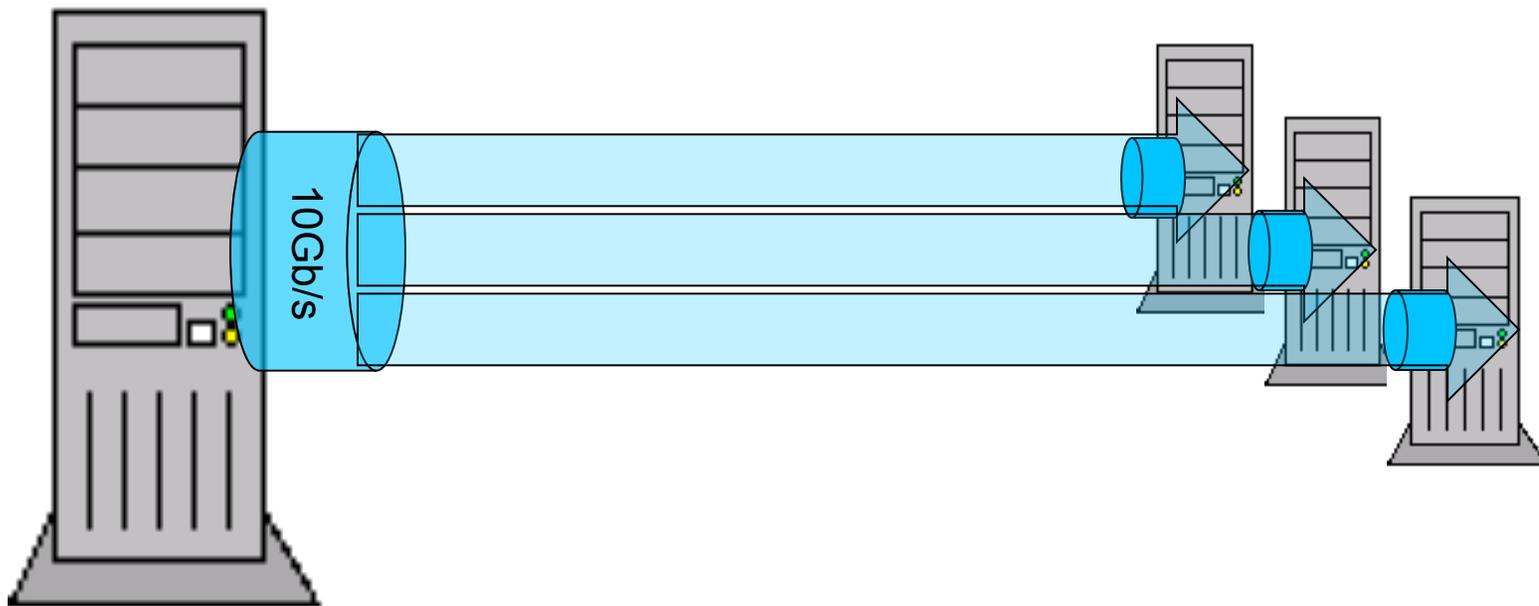
Simultaneous Sessions

- Goal: Collective throughput
 - ◆ entire servers bytes transferred / time
 - Not the number of transfers at once
- Only reasons for more than 1 connection
 - ◆ Provide an interactive service for many
 - ◆ One session does not use all of the local resource
 - The remote side is the bottleneck
 - ◆ Hide control messaging overhead in another sessions data transfer payload



Remote Bottleneck

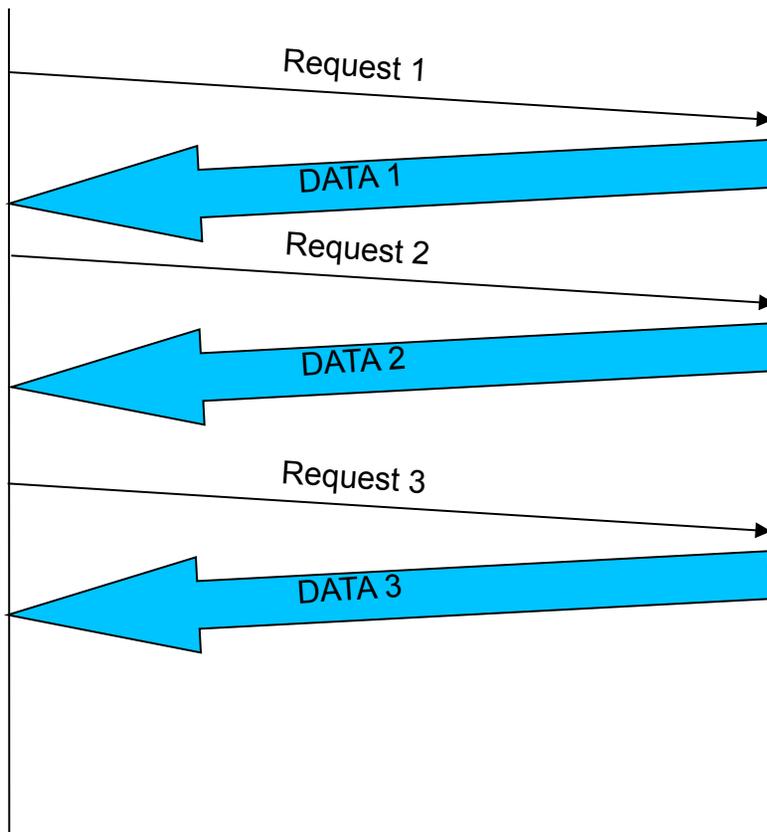
- Allow more than one simultaneous transfer to use all resources



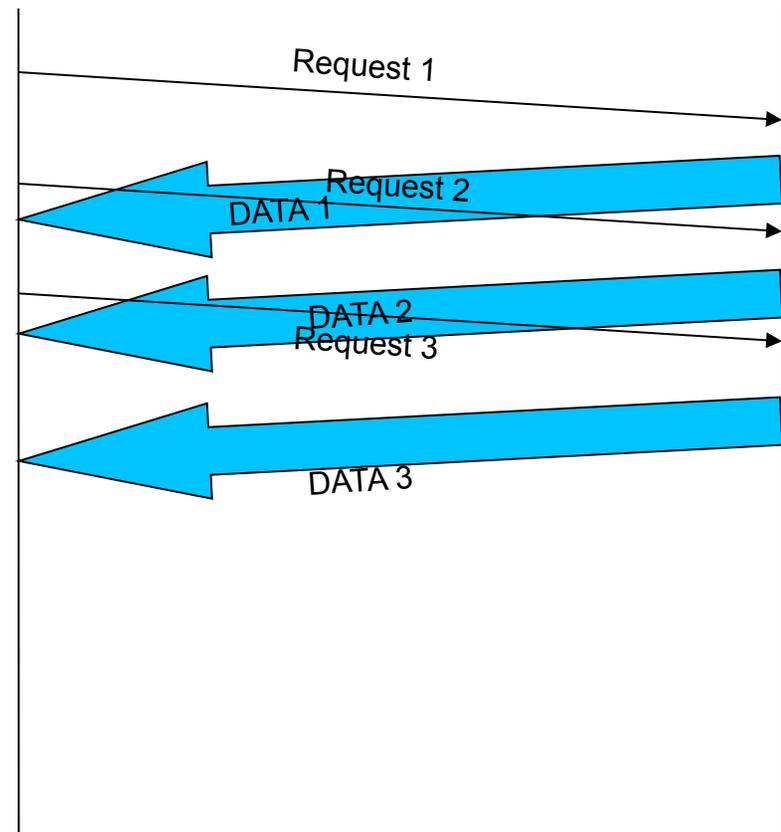


Overhead hiding

Single Connection



Many Connections

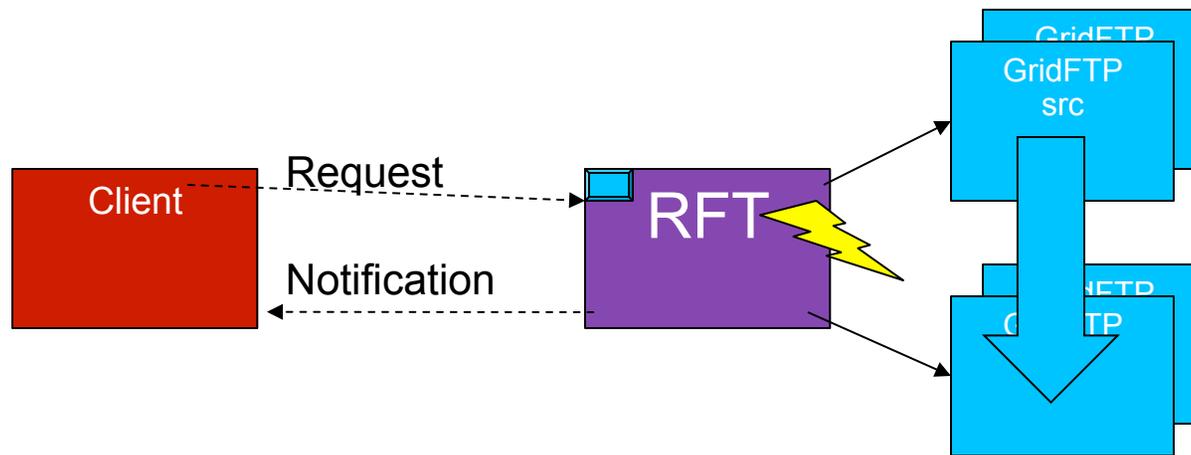




But We Want Queuing!

- May I offer you something in an RFT?
 - ◆ RFT says yes
 - ◆ Server side retries
- Light weight sessions
 - ◆ GridFTP does the heavy lifting
 - ◆ Queues up requests of pending transfers
 - ◆ Notification upon completion
 - ◆ Scalability
- Manages/Optimizes access to GridFTP Servers

RFT Session Interactions



Scalability

- GridFTP

- ◆ Connection rejection is a feature
 - It SHOULD say no
- ◆ Intended to scale to system transfer rates
 - Not beyond them
 - T
O
S
C
scale up add more nodes as stripes (dynamic backbends)
 - Use faster NICs

- RFT

- ◆ Intended to scale to memory
 - It should not say no



GridFTP Broke My Cluster!

- GridFTP will push hardware as hard as it is allowed
 - ◆ But not harder
- `sudo rm -rf /`
 - ◆ Did sudo break the FS?
- `ssh -u root host1 fork.bomb`
 - ◆ Did sshd take down the host?
- `globus-url-copy -tcp-bs 100GB <src> <dst>`
 - ◆ Did GridFTP break the cluster?



Resource Protection

- Limits need to be in place to protect
 - ◆ Knowing it is ok to say 'no' is step 1
- What will hardware allow?
 - ◆ How fast are my disks?
 - ◆ How fast is my NIC?
 - ◆ How fast is can I send data while using the NetFS?
 - ◆ How many WAN transfers can I support with system memory?
 - ◆ How many simultaneous transfers can are reasonable to sustain?

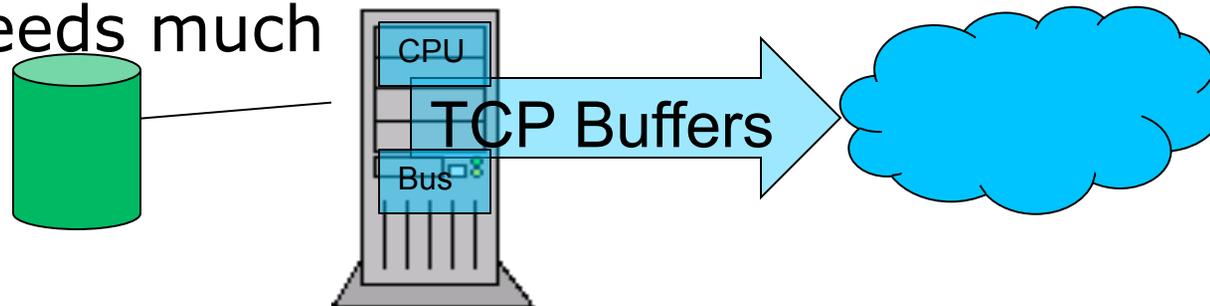


Fast Transfer Resources

- CPU
 - ◆ Packet switching
- Memory
 - ◆ OS buffers (BWDP)
 - ◆ User space buffers
 - ◆
- System bus
- Disk
 - ◆ Shared FS? (net also)
- Network
 - ◆ Router and LAN

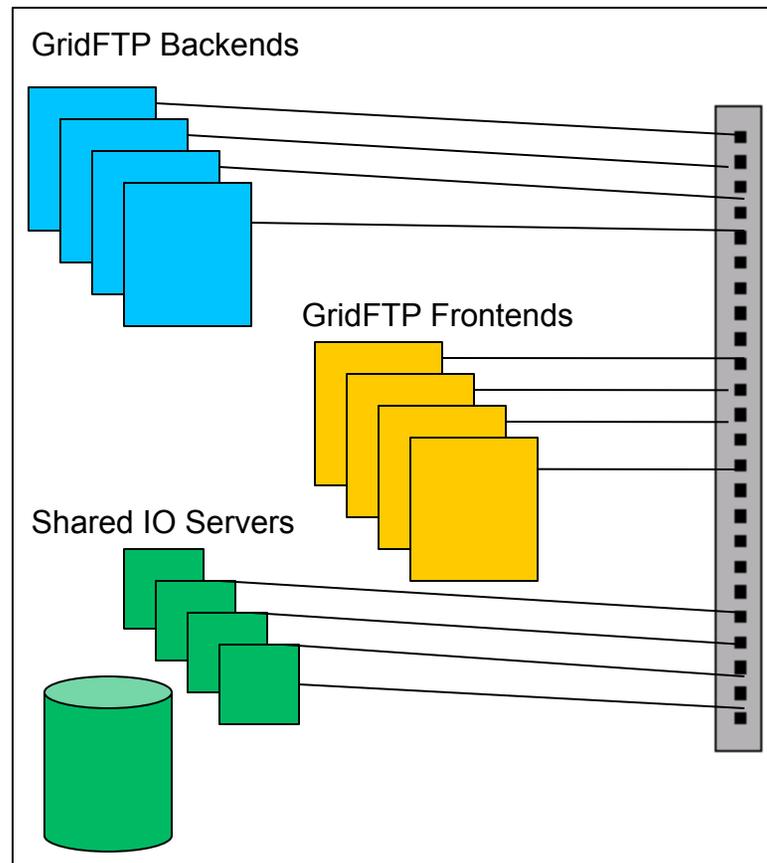
W
A

N needs much





Cluster Components



- Disk
 - ◆ Shared I/O servers
- Net
 - ◆ Backplate bandwidth
- Systems
 - ◆ CPU/Memory
 - ◆ Are IODs and GridFTP servers co located?



Connection Caps

- As a function of system memory
 - ◆ $\text{Cap} = \lfloor \text{mem} \rfloor / (2\text{MB} + \text{avg}(\text{BWDP}))$
 - ◆ Never more than $\lfloor \text{mem} \rfloor / 4\text{MB}$

```
service gsift
{
    instances          = 20
    socket_type        = stream
    wait                = no
    env                 += GLOBUS_LOCATION=...
    env                 += LD_LIBRARY_PATH=...
    server              = /usr/local/globus-4.0.1/sbin/globus-gridftp-server
    server_args         = -i -p 2811
    disable             = no
}
```

```
% globus-gridftp-server -connection-max 20
```



Connection Caps

- As a function of system bandwidth
 - ◆ Cap =
 $\min(\text{FS. B}, \text{Net.BW}) / (\text{Target average transfer rate})$
- As a function of my gut
 - ◆ 20 - 50
 - ◆ Best guess based on personal experience
 - ◆
T
ypically this is where collective BW plateaus



System Buffer Limits

- Limit the amount of OS space per connection
 - ◆ Auto tuning
 - ◆ 16MB - 64MB

```
% sysctl -w net.core.rmem_max=<value>  
% sysctl -w net.core.wmem_max=<value>
```

```
% cat /proc/sys/net/ipv4/tcp_wmem  
4096 16384 4194304
```

```
% cat /proc/sys/net/ipv4/tcp_rmem  
4096 16384 4194304
```



GFork Memory Manager

- Dynamically rations memory
 - ◆ 10% of the allowed connections get 90% of the memory
 - ◆ Remaining session get half of available memory
- Allows for high connection limits
 - ◆ |mem| / 2MB



Future Work

- RFT Improvements
 - ◆ Observe and react to GridFTP workloads
 - Current transfer rates
 - Requested TCP buffer sizes
- Dynamic connection limits
 - ◆ More GForK memory algorithms
 - ◆ Base on current throughput
- Queuing Service
 - ◆ Mainly for use by RFT
 - ◆ Eliminates possible starvation
- Formal Study



Conclusions

- GridFTP is an on demand service
 - ◆ OK to say no
- RFT is a VO level queuing service
 - ◆ please use it
- <http://www.gridftp.org>
- gridftp-user@globus.org