Bulk Data Transfer Techniques for High-Speed Wide-Area Networks

Brian L. Tierney
ESnet

http://fasterdata.es.net/
http://dsd.lbl.gov/TCP-tuning/
Why does the Network seem so slow?
Wizard Gap

Slide from Matt Mathis, PSC
This talk will cover:

- Some Information to help you become a “wizard”
- Work being done so you don’t have to be a wizard

Goal of this talk:

- Help you fully optimize wide area bulk data transfers
  - or help your users do this

Outline

- TCP Issues
- Bulk Data Transfer Tools
- Network Monitoring Tools
- New TCP Stacks
  - how they help with, but not eliminate, the “wizard gap”
Time to Copy 1 Terabyte

- 10 Mbps network: 300 hrs (12.5 days)
- 100 Mbps network: 30 hrs
- 1 Gbps network: 3 hrs
- 10 Gbps network: 20 minutes
  - need fast disk array for this

Compare these speeds to:
- USB 2.0 portable disk
  - 60 MB/sec (480 Mbps) peak
  - 20 MB/sec (160 Mbps) reported on line
  - 5-10 MB/sec reported by colleagues
  - 15-40 hours to load 1 Terabyte
Sample Buffer Sizes

• LBL to (50% of pipe)
  – SLAC (RTT = 2 ms, narrow link = 1000 Mbps) : 256 KB
  – BNL: (RTT = 80 ms, narrow link = 1000 Mbps): 10 MB
  – CERN: (RTT = 165 ms, narrow link = 1000 Mbps): 20.6 MB

• Note: default buffer size is usually only 64 KB, and default maximum autotuning buffer size for is often only 256KB
  – Linux/FreeBSD Autotuning default max = 256 KB (recently increased to 1 MB);
  – 10-150 times too small!

• Home DSL, US to Europe (RTT = 150, narrow link = 2 Mbps): 38 KB
  – Default buffers are OK.
• To solve the buffer tuning problem, based on work at LANL and PSC, Linux OS added TCP Buffer autotuning
  – Sender-side TCP buffer autotuning introduced in Linux 2.4
  – Receiver-side autotuning added in Linux 2.6
• Many other OS’s now include TCP autotuning
  – TCP send buffer starts at 64 KB
  – As the data transfer takes place, the buffer size is continuously re-adjusted up max autotune size
• Current OS Autotuning default maximum buffers
  – Linux 2.6: 256K to 1MB, depending on version
  – FreeBSD 7: 256K
  – Windows Vista: 16M
  – Mac OSX 10.5: 8M
Autotuning Settings

- **Linux 2.6**
  
  ```
  net.core.rmem_max = 16777216
  net.core.wmem_max = 16777216
  # autotuning min, default, and max number of bytes to use
  net.ipv4.tcp_rmem = 4096 87380 16777216
  net.ipv4.tcp_wmem = 4096 65536 16777216
  ```

- **FreeBSD 7.0**
  
  ```
  net.inet.tcp.sendbuf_auto=1
  net.inet.tcp.recvbuf_auto=1
  net.inet.tcp.sendbuf_max=16777216
  net.inet.tcp.recvbuf_max=16777216
  ```

- **Windows Vista**
  
  ```
  netsh interface tcp set global autotunninglevel=normal
  ```
  - max buffer fixed at 16MB

- **OSX 10.5 (“Self-Tuning TCP)**
  
  ```
  kern.ipc.maxsockbuf=16777216
  ```

- **For more info, see:** [http://acs.lbl.gov/TCP-Tuning/](http://acs.lbl.gov/TCP-Tuning/)
Parallel Streams Results with various TCP window sizes

RTT = 70 ms

Graph from Tom Dunigan, ORNL
Tuned Buffers vs. Parallel Steams

Throughput (Mbits/sec)

- no tuning
- tuned TCP buffers
- 10 parallel streams, no tuning
- tuned TCP buffers, 3 parallel streams
• Potentially unfair
• Places more load on the end hosts
• But they are necessary when you don’t have root access, and can’t convince the sysadmin to increase the max TCP buffers

graph from Tom Dunigan, ORNL
Summary so far

• To optimize TCP throughput, do the following:
  – Use a newer OS that supports TCP buffer autotuning
  – increase the maximum TCP autotuning buffer size
  – use a few parallel streams if possible

• But also, try to ‘play nice’:
  – Leave some bandwidth for others.
  – e.g.: Don’t try to completely fill your networks 1000BT uplink
  – Good ‘rule of thumb’:
    • Try for 100-200 Mbps per stream, and use around 4 streams
Bulk Transfer Tools
Sample Data Transfer Results: LBL to BNL

• Using the right tool is very important
  – scp / sftp : 10 Mbps
    • standard Unix file copy tools
    • fixed 64 KB TCP window in openSSL
  – ftp : 400-500 Mbps
    • assumes TCP buffer autotuning
  – parallel stream FTP: 800-900 Mbps
• Don’t use scp to copy large files across a WAN!
  – scp has its own internal 64KB buffering/windowing that prevents it from ever being able to fill LFNs!

• Explanation of problem and openssh patch solution from PSC:
  – http://www.psc.edu/networking/projects/hpn-ssh/
• Uses libopenssl, so don't use sftp WAN transfers unless you have installed the HPN patch from PSC
• But even with the patch, SFTP has yet another flow control mechanism
  – By default, sftp limits the total number of outstanding messages to 16 32KB messages.
  – Since each datagram is a distinct message you end up with a 512KB outstanding data limit.
  – You can increase both the number of outstanding messages ('-R') and the size of the message ('-B') from the command line though.

• Sample command for a 128MB window:
  – sftp -R 512 -B 262144 user@host:/path/to/file outfile
• GridFTP from ANL has the ability to fill the network pipe
  – Buffer Tuning
  – Parallel Streams
• Supports multiple authentication options
  – anonymous
  – ssh (available in starting with Globus Toolkit version 4.2)
  – X509
• Ability to define a range of data ports
  – helpful to get through firewalls
• Sample Use:
  – `globus-url-copy -p 4 sshftp://data.lbl.gov/home/mydata/myfile`
  – `file://home/mydir/myfile`
• Available from: http://www.globus.org/toolkit/downloads/
new GridFTP Features

• New ssh authentication option
  – Not all users need or want to deal with X.509 certificates
  – Solution: Use SSH for Control Channel
    • Data channel remains as is, so performance is the same
  – see http://fasterdata.es.net/gridftp.html for a quick start guide

• Optimizations for small files
  – Pipelining: for cases where many transfer requests are outstanding at once
    • Client sends next request before the current completes
    • Latency of request is hidden in data transfer time
  –Cached Data channel connections
    • Reuse established data channels (Mode E)
    • No additional TCP or GSI connect overhead

• Support for UDT protocol
new GridFTP “bottleneck detector”

- new command line option for globus-url-copy, "-nlb"
  - nlb = NetLogger bottleneck
  - Uses NetLogger libraries for analysis of network and disk I/O
    - http://acs.lbl.gov/NetLogger

- Possible "Bottleneck:" results are:
  - network: somewhere in the network
  - disk read: sender's disk
  - disk write: receiver's disk
  - unknown: disk/network are about the same and/or highly variable
new GridFTP “bottleneck detector”

• Sample Output:

Total instantaneous throughput:
disk read = 1235.7 Mbits/s
disk write = 2773.0 Mbits/s
net read = 836.3 Mbits/s
net write = 1011.7 Mbits/s

Bottleneck: network

• instantaneous throughput is the average # of bytes divided by the time spent blocking on the system call.

• instantaneous throughputs are higher than the overall throughput of the transfer:
  – does not include the time waiting for data to be available
  – primarily useful for comparison and not as absolute numbers

• Ignore the "net write" value
  – the time to write to the network is strongly influenced by system and TCP buffer artifacts.
new GridFTP “bottleneck detector”

- -nlb not enabled by default
  - use ./configure --enable-netlogger
  - additional server configuration flags needed
  - instructions at:
Phoebus

• See Martin Swany’s Talk, Tuesday, 3:20pm

http://e2epi.internet2.edu/phoebus.html
• bbftp (from the HEP “Babar”) project also everything needed to fill the network pipe
  – Buffer Tuning
  – Parallel Streams
• Supports ssh authentication options
• Supports on-the-fly compression
• Sample Use:
  – `bbftp -p 4 bbftp://data.lbl.gov/home/mydata/myfile`
    `file:///home/mydir/myfile`
• Available from:  http://doc.in2p3.fr/bbftp/
• bbcp: http://www.slac.stanford.edu/~abh/bbcp/
  – supports parallel transfers and socket tuning
  – bbcp -P 4 -v -w 2M myfile remotestore:filename

• lftp: http://lftp.yar.ru/
  – parallel file transfer, socket tuning, HTTP transfers, and more.
  – lftp -e 'set net:socket-buffer 4000000; pget -n 4 [http|ftp]://site/path/file; quit'

• axel: http://wilmer.gaast.net/main.php/axel.html
  – simple parallel accelerator for HTTP and FTP.
  – axel -n 4 [http|ftp]://site/file
Download Managers

• There are a number of nice browser plugins that can be used to speed up web-initiated data transfers
  – all support parallel transfers

• Firefox add-on (All OSes):
  – DownThemAll: http://www.downthemall.net
  – this is my favorite: probably the best/simplest solution

• For Linux:
  – aria: http://aria-rpm.sourceforge.net

• For Windows:
  – FDM: http://www.freedownloadmanager.org
  – Stardownload: http://www.stardownload.com/

• For OSX:
  – Speed Download: http://www.yazsoft.com/ ($25)
• **Open Source:**
  - http://filezilla-project.org/
  - includes client and server

• **Features:**
  - ability to transfer multiple files in parallel

• **Issues:**
  - uses libopenssl in secure mode, so has buffer limitations
Special Purpose Data Transfer Tools

• HPSS Tools: HSI and pftp
  – both support buffer tuning and parallel transfers
    • per destination buffer tuning must be done by HPSS admin

• SRM from SDSC
  – supports buffer tuning and parallel transfers
Selecting a bulk data transfer tool

• First, determine which security model you require
  – anonymous: (e.g.: FTP, HTTP) anyone can access the data
  – simple password: (e.g.: FTP, HTTP) most sites no longer allow this method since the password can be easily captured
  – password encrypted: (e.g.: bbcp, bbftp, GridFTP) control channel is encrypted, but data is unencrypted
  – everything encrypted: (e.g.: scp, sftp, GridFTP, HTTPS-based web server) both control and data channels are encrypted

• Most open science projects seem to prefer option #3.

• If you require option #4, tools that perform well over a WAN are limited to:
  – GridFTP with X509 keys,
  – HPN-patched versions of scp/sftp.
TCP Congestion Control Issues
Recent TCP Congestion Control Modifications

• Many Proposed Solutions:
  – High Speed TCP: Sally Floyd
    • http://www.icir.org/floyd/hstcp.html
  – BIC/CUBIC: (CUBIC now on by default in Debian/Ubuntu)
    • http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/
  – LTCP (Layered TCP)
    • http://students.cs.tamu.edu/sumitha/research.html
  – HTCP: (Hamilton TCP)
    • http://www.hamilton.ie/net/htcp/
  – Scalable TCP
    • http://www-lce.eng.cam.ac.uk/~ctk21/scalable/
  – CTCP (Compound TCP) (available, off by default, in Vista)
Linux 2.6.12 Results

Note that BIC reaches Max throughput MUCH faster.

TCP Results

RTT = 67 ms
Comparison of Various TCP Congestion Control Algorithms

- StandardTCP from SLAC to Florida
- HSTCP from SLAC to Florida
- CTCP from SLAC to Florida
Selecting TCP Congestion Control in Linux

• To determine current configuration:
  
  `sysctl -a | grep congestion`

  ```
  net.ipv4.tcp_congestion_control = cubic
  net.ipv4.tcp_available_congestion_control = cubic
  reno
  ```

• Use `/etc/sysctl.conf` to set to any available congested congestion control.

• To enable CTCP in Vista, set the following:
  
  `netsh interface tcp set global congestionprovider=ctcp"`
More Information

• http://fasterdata.es.net/

• http://acs.lbl.gov/TCP-tuning/

• email: BLTierney@es.net
Extra (advanced) slides