University of California, Santa Cruz

Large Genomic Data Transfer Methods

GNET: Project Update

HTTP://GNET.SOE.UCSC.EDU

Authors:

Manikandan Punniyakotti

MANI@UCSC.EDU

Sam Wood

SBWOOD@UCSC.EDU

May 6, 2011

1. Introduction

As genome sequencing becomes cheaper and more frequent there is a rising demand amongst bioinformatics researchers for a quick method of transferring large datasets over long distances for collaboration. For example, the *1000 Genomes Project* provides a database of freely available sequenced genomes with the aim of promoting research in understanding disease through genotype variation. The complete human genome in the KOREF_20090131 dataset is 3.0GB in size, without metadata or annotations (FASTA format, roughly 892MB using gzip compression). Note that annotated data that includes metadata such as sequence alignment (SAM or BAM format) commonly increases the file size by factors greater than 4.

Data transfers of dozens of genomes between data centers on opposite sides of the United States are daily occurrences. This paper examines genomic data transfer between hosts on a high speed network such as Internet2, with long round trip times (RTTs); this type of connection is known as a long fat network (LFN) due to the large bandwidth delay product. A typical link would have an RTT of roughly 100ms, 15 hops, and a capacity of 10Gbps. It is known that default TCP buffer sizes and congestion avoidance algorithms are not tuned for such networks, nor are the typical application file transfer protocols that use TCP, such as ftp, http, and scp.

The goal of this paper is to provide a set of guidelines to end hosts of a large genomic data transfer that will reduce the total transmission duration (compared to existing methods). To be sure, this paper's scope is limited to changes on the end hosts and assumes an immutable intermediate network. Secondary goals include providing secure encryption and network fairness. We justify these guidelines through extensive simulation, with realistic parameters from network traces. Specifically, we use the Dummynet network emulator to model a LFN with characteristics identical to that of several Internet2 paths used in current large genomic

data transfers. Also, we examine 3 large data transfer applications: GridFTP, FDT, and paraFetch, along with their respective optimal TCP settings.

Lastly, we comment on novel content-specific data compression techniques for genomic data. Indeed, we argue that the best way to reduce data transfer duration is to reduce the amount of data sent over the network. We conclude with future work that exploits the redundant nature inherent to genomic data, as well as possible network changes that could further reduce transfer duration.

1.1 Contents

Chap. 2 gives a brief background and commentary to the tools and practices currently used by data centers to transfer large datasets across high capacity, high RTT links.

Chap. 3 details a test plan and a discussion about the relevance of the test plan's results in genomic data distribution. Additionally, we provide preliminary results for several network scenarios.

Chap. 4 summarizes the deliverables and what remains as further research. In particular, we comment on the role of content-specific compression for genomic data in data transfer.

2. Background

TCP is very popular for unicast communications and has been packaged with commodity operating systems and networking APIs. Hence TCP is widely used by networking applications over a variety of networking media.

However, for long-haul high bandwidth networks (commonly called Long Fat Networks) commodity TCP has been found to be less suitable. This disadvantage is because even if the link is slightly error-prone, TCPs conservative congestion control mechanisms reduce the throughput heavily by underutilizing the large bandwidth delay product. Also, TCP provides reliability through ACKS and retransmissions and hence the latency of a packet recovery is at least an RTT of such long delay links. Also, TCP requires huge buffers at the end hosts to fully utilize the capacity. To perform efficient bulk data transfers over such networks, without changing the underlying network architecture and with TCP at the transport layer, several approaches such as tuning the TCP parameters at the end-hosts, using better congestion control algorithms in place of the traditional TCP Reno, and using sophisticated data transfer tools are employed.

2.1 TCP Tuning

TCP tuning generally refers to adjusting the TCP buffers that correspond to the TCP windowing mechanism. Most applications do not try to understand the network in detail, nor learn the distance to the other end of the communication. A solution to this oversight is TCP auto-tuning with pre-configured limits. Sender-side auto-tuning was introduced in Linux 2.4 while receiver-side support was added in Linux 2.6. However, some of the default values that are used for auto-tuning still are not optimized for LFNs.

Fasterdata (ESnet 2011) is a knowledge base dedicated to informing network administrators on how to transfer large (hundreds of gigabytes to terabytes)

datasets over LFNs. Fasterdata is part of the Energy Sciences Network (ESnet) a high-speed network serving the United States Department of Energy.

Fasterdata suggests several changes that need to be made to the Linux TCP kernel settings, typically stored in "/etc/sysctl.conf", to improve TCP auto-tuning. For TCP max buffer size, 16MB is recommended for most 10Gbps paths and for some very long RTT, 10Gps or 40Gps paths, 32 MB is suggested (see Fig. 2.1).

```
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
```

Figure 2.1: Recommended TCP max buffer sizes for LFNs

The auto-tuning "maximum TCP buffer" limits should be changed to 16MB as well, while leaving the minimum and default TCP buffer sizes as their defaults (see Fig. 2.1).

```
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
```

Figure 2.2: Recommended TCP auto-tuning buffer sizes for LFNs

Fasterdata recommends a value of 30000 for the receiver's incoming packet backlog queue. [1]

Linux supports pluggable congestion control algorithms. Fasterdata recommends CUBIC (Ha, Rhee & Xu 2008) or HTCP be used, since they do not rely on RTT values for adjusting the congestion window sizes. [2] Fasterdata recommends NIC tuning by modifying "/etc/rc.local" to load at the settings boot time. Specifically, they recommend a transmission queue length of 10000 for 10Gbps NIC cards.

[1] the "net.core.netdev_max_backlog" parameter

[2] the "net.ipv4.tcp_available_congestion_control" parameter

SpeedGuide (SpeedGuide.net 2011) is a website dedicated to informing network administrators on improving Broadband Internet performance. They recommend enabling selective ackowledgements, enabling TCP window scaling (to allow window sizes to exceed 65535), and disabling timestamps (with the caveat that some congestion control protocols require accurate timestamps). See Fig. 2.1.

```
net.ipv4.tcp_sack = 1
net.ipv4.tcp_window_scaling = 1
net.ipv4.tcp_timestamps = 0
```

Figure 2.3: Recommended SpeedGuide.net TCP settings

2.1.1 perfSONAR

PerfSONAR is an infrastructure for network performance monitoring, making it easier to solve end-to-end performace problems on paths crossing several networks. It composes of several network monitoring tools, including: BWCTL (Bandwidth Test Controller) that can use Iperf, Thrulay or Nuttcp; OWAMP (One Way Ping); NDT (Network Diagnostic Tool); ping and traceroute.

As a part of this project, we setup a perfSONAR node at UCSC to monitor the paths to genomic data destinations around the Internet. We studied the characteristics of these paths and used them to configure our Dummynet testbed so that our experiments yield realistic and accurate results. The following are the parameters we obtained through perfSONAR and its tools:

1. RTT
2. Packet loss rate
3. End-end Throughput
4. Number of hops

2.2 Data Transfer Tools

The tools that are used for data transfer play a vital role in minimizing the total end-to-end transmission duration. Tools that support transfers through parallel streams are needed as it is easier to achieve a given performance level through multiple parallel connections than through one connection. Also, WAN transfers have much higher latency than LAN transfers but many tools like SCP or SFTP assume a LAN. Thus using the right tool is essential to efficiently do bulk-transfers.

2.2.1 Fast Data Transfer (FDT) Application

FDT is an application to do efficient data transfers over wide area networks with standard TCP. It is capable of reading and writing at disk speed over such networks. This tool is java based, and can run on all major platforms. FDT uses the capabilities of the Java NIO libraries and is based on an asynchronous, flexible multithreaded system. It supports several features including parallel data transfer, resuming a file transfer session without loss, and continuous streaming of a list of files using a managed pool of buffers. A large set of files can be sent and received at full speed without having to restart the network traffic between files.

This tool is simple to install and use: FDT has a server and client that are included in a single JAR file and two scripts have been provided to run the server and the client separately. Once the JAR file is placed on both hosts, and the server is started using the provided script, the tool is ready for use: just start the client script with appropriate options. Also, installation and operation does not require administrative privileges.

2.3 GridFTP Application

GridFTP is an extension of the standard FTP protocol and it is defined as part of the Globus toolkit. The Globus Alliance develops Grid technology to make resource management, security, and data management standardized and straightforward. GridFTP was developed to provide a more reliable and high performance file transfer for Grid computing applications that needed to transmit very large files quickly and reliably.

GridFTP includes features such as: security with Grid Security Infrastructure (GSI), third party transfers (a local client can initiate remote transfers between servers), parallel and striped transfer, mutliple source to single destination transfer, partial file transfer (can be resumed from a specific point or transmission of just a subset of a file), fault tolerance and restart (can handle unavailability or sever problems and automatically restart after a problem), automatic TCP optimization (negotiation of TCP buffer sizes and window sizes to provide transfer speeds and reliability), data port range (to allow working around firewalls), and UDT support.

2.4 paraFetch Application

paraFetch is a bulk data transfer tool developed in-house at UCSC for Genome data transfers between data centers. It is entirely written in C and compiling the source code and installing is relatively difficult. Hence, in our experiments we used the binary files directly as they were compatible with the machines we were using for our experiments. paraFetch tool fetches files behind a webserver and hence we need to setup paraFetch only on the client side provided we have a webserver already setup on the server side that hosts the files to be transferred. paraFetch supports both HTTP and HTTPS protocols. For our experiments we used nginx, a lightweight webserver that supports both HTTP and HTTPS protocols and does not require root to install. paraFetch uses 50 parallel TCP streams, by default, to fetch the data in parallel; this number can be increased.

2.5 Aspera Note

[[TODO: talk about Aspera methodology, easy of use, anecdotal results, last update]]

2.6 UDP Note

[[TODO: talk about UDP blasters, are they effective? (facebook thinks so)]]

3. Scenarios

The main metric of interest is total end-to-end transmission duration, with the goal of minimizing this metric. A secondary goal is maintaining network fairness: an optimal solution should be fast but not introduce packet loss in existing sessions. We used packet loss in non-genomic TCP data transfer sessions during genomic data transfer as a measure of fairness. Lastly, the solution must be compatible with some form of encryption, whether it is built-in or uses an existing tool such as SSH.

In order to limit the scope of this paper, we examine two Internet2 network configurations that are currently used in genomic data transfer: UCSC to the the Baylor College of Medicine in Houston, Texas (case A); and UCSC to the Broad Institute (case B) in Massachusetts. These two cases were picked since all three participate in genomic data transfer and their geographical distances result in a medium range RTT of roughly 42ms, and a long range RTT of 92ms, respectively. One would expect many genomic data transmission within the United States over Internet2 to fall within these ranges.

The following are parameters that are dependent on the particular scenario, but remain fixed throughout our experiments for each scenario: RTT, packet loss, throughput and number of hops. Other characteristics such as jitter are important for ordered time sensitive data, whereas in large data transfer they have less of an influence on transmission duration.

Within each scenario, we varied the following factors in a full-factorial experiment: solution application (GridFTP, paraFetch, FDT), number of parallel streams, host TCP settings, with or without encryption, and memory-to-memory versus disk-to-disk transfers.

Due to an inability to run in-depth experiments on end hosts directly, and for reproducibility, we modeled the network for these two scenarios using the Dum-

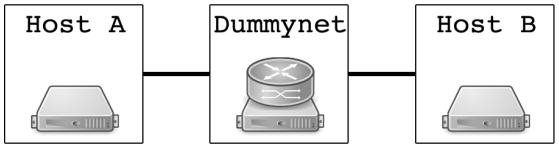

Figure 3.1: Testbench configuration

mynet network emulator (see Fig. 3.1), with three identical hosts. [1] Specifically, the RTT, capacity, number of hops, and packet loss parameters for Dummynet were set for the corresponding scenario using data collected from perfSONAR (see 2.1.1) between UCSC and the University of Texas (for case A), and between UCSC and the M.I.T. Lab for Nuclear Sciences (for case B). We were unable to find perfSONAR hosts directly at the Baylor College of Medicine nor the Broad Institute. Another caveat is that we used 1Gbps Ethernet throughout all of our scenarios, when in reality the Internet2 backbone supports up to 40Gbps. To be sure, most end hosts are connected to a switch with a lower rate.

`traceroute` was used to calculate the number of hops and intermediate node RTTs from UCSC to the corresponding destination. Dummynet allows multi-hop configurations with varying delays, but only at a millisecond granularity. Note that using `traceroute` to estimate RTTs between intermediate hops is not necessarily precise, as the RTTs include fluctuating queuing delays and are a bidirectional metric. Fig. 3.2 lists the Dummynet two-way propagation delays side-by-side with the delays calculated using `traceroute`; note that the total Dummynet RTT is within two milliseconds of the actual RTT. Additionally, the Dummynet delay is the two-way propagation delay and does not include queueing delays which can vary from system to system.

3.1 Preliminary Results

Fig. 3.3 details the achieved throughput using the 3 different tools in Case A, where higher throughput is better. For this particular experiment, we did a memory-to-memory transfer of a 3.7GB FASTA human genome. We used the

[1] 2 Processor AMD Opteron(tm) Processor 246 HE, 8GB RAM, 1Gbps Ethernet

hop#	Est. RTT Delay(ms)	Dummynet Two-way Prop. Delay(ms)
1	0.173	0
2	1.062	0
3	7.983	4
4	0.151	0
5	30.843	14
6	0.254	0
7	0.100	0
8	2.116	2
9	0.100	0
total RTT (ms)	42.782	41 (20+queuing delay)

Figure 3.2: Case A: Dummynet pipe delays compared to actual RTTs

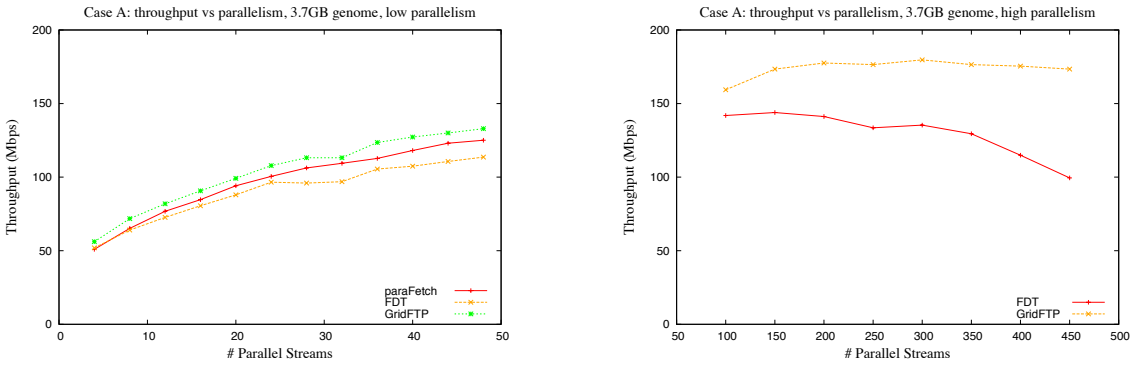

Figure 3.3: Case A: memory-to-memory, varying parallelism

TCP tuned settings described in Sec. 2.1 and disabled encryption. Note that paraFetch has a limitation of 50 parallel streams and the two graphs demonstrate performance on a small and large number of parallel streams. GridFTP consistently achieves higher throughput in this scenario, and hence lower end-to-end transmission duration.

4. Conclusion

[[TODO: detail specific guidelines based on results]]

4.1 Content-Specific Compression

An alternative approach to reducing end-to-end transmission duration is to reduce the amount of data transmitted using preprocessing such as content-specific compression. Indeed, (Wang & Zhang 2011) details a novel reference method of compression for genomic data that can reduce a 2986.8MB FASTA file to under 18.8MB by first applying Huffman encoding and then storing the differences (using a modified version of the Unix `diff` tool) between a reference genome and the genome to be compressed. This method could be applied for genomic data transfer by having clients first receive a large reference genome and then receive the differences for the subsequent genomes. To be sure, such a method is more CPU intensive than the current method of hosting gzipped files, however the CPU cost can be amortized across delivering the file to multiple clients (the compression is only computed once for each new genome). The possible bandwidth savings using this type of compression is immense. More research is needed to explore the benefits of referenced based and content specific compression with respect to network transfer.

Other relevant papers: (Balakrishnan 2009), (Mathis, Heffner & Reddy 2003), (Greenberg, Hamilton, Jain, Kandula, Kim, Lahiri, Maltz, Patel & Sengupta 2009)

References

Balakrishnan, M. (2009), Reliable Communication for Datacenters, PhD thesis, Cornell.

ESnet (2011).

Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P. & Sengupta, S. (2009), 'Vl2: a scalable and flexible data center network', *SIGCOMM Comput. Commun. Rev.*

Ha, S., Rhee, I. & Xu, L. (2008), 'Cubic: a new tcp-friendly high-speed tcp variant', *SIGOPS Oper. Syst. Rev.*

Mathis, M., Heffner, J. & Reddy, R. (2003), 'Web100: extended tcp instrumentation for research, education and diagnosis', *SIGCOMM Comput. Commun. Rev.*

SpeedGuide.net (2011).

Wang, C. & Zhang, D. (2011), 'A novel compression tool for efficient storage of genome resequencing data', *Nucleic Acids Research*.