Access to Big Data in Bioinformatics

Andrew van Rooyen
University of Cape Town
Andrew.vanRooyen@alumni.uct.ac.za

ABSTRACT

TODO

1. INTRODUCTION

Next generation sequencing has resulted in a massive increase in the size of bioinformatics datasets, which can be tens of gigabytes in size [5]. Sequencing technologies like SOLiD provide much higher data output at a cheaper cost [15], which creates challenges in data storage, transfer and access. In fact, the cost of storing a byte has been higher than sequencing a base pair since before 2010 [2].

Generally, sequence data is stored in a data warehouse. Storing this information for long periods of time requires the data to be structured efficiently in order to save space and to allow it to be transferred efficiently.

There are a plethora of sequence file formats whose efficiency depends on the kind of data stored. Two of the most popular are FASTQ, which stores aggregated reads along with the quality of each DNA base pair [4], and BAM, the binary, compressed version of the Sequence Alignment Map (SAM) format [13].

Researchers often require access to the data warehouses to transfer the sequences they need. Luckily, these locations are often connected by massive data pipes like National Research and Education Networks (NREN's). For example, South African universities are connected by the South African National Research Network [14] which runs at 10Gbps. Unfortunately, standard transfer protocols like FTP and SSH were not designed for use on high-throughput networks, and alternate protocols must be used to avoid bottlenecks.

Some proprietary transfer protocols are widely used in practice for example, the *fasp* protocol by the US based company AsperaSoft. Based on UDP, this protocol eliminates the latency issues seen with TCP, and provides transfer bandwidth of up to 10 gigabits per second [6].

There have been some attempts to avoid data transfer altogether. This means processing data remotely, and there has been an explorative push towards cloud solutions from companies such as Amazon and Google [2]. Unfortunately, even though cloud data centres have plenty of cheap storage, the transfer bottleneck remains as researchers must still upload their raw data to the cloud data centres every time they run a new experiment. Some researchers have even resorted to mailing hard drives [2].

There are also security, privacy and ethical concerns with outsourcing processing power to other companies, as sequenced DNA data is often highly sensitive information [9].

Although it would be ideal for researchers if this reliance on data transfer was removed, the current solutions do not provide good enough answers at every scale. Therefore, transferring big data is still a necessary evil for the time being.

In situations where data *must* be transferred, it stands to reason that the transfer should happen in the most efficient way.

In this work, we construct a testbed which can be deployed on a network to test various transfer protocols. The testbed is outlined, and then used to test SCP, FTP, GridFTP and HPN-SCP. These are the most popular protocols for file transfer in the field, perhaps with the exception of AsperaSoft's *fasp*, which is non-free.

In order to test the protocols in a relevant environment, the tests are run between the University of Cape Town (UCT) and the University of the Western Cape (UWC) which are connected via SANReN.

2. METHODS

A testbed is created to measure speed, data efficiency and packet size of specific transfer protocols. Transfers are run and information about them is captured and saved. These logs are then analysed and displayed visually. The testbed does not try to come up with the 'best transfer protocol', but rather presents the information relevant to the specific environment so that the user can make decisions.

2.1 Testbed Design

The testbed has been designed to be simple, modular and extensible. A simple Python system is sufficient, because almost all of the work transferring and logging the data is done by external programs. The testbed simply acts as a mediator between them.

The data dump format is a simple json template, and can be parsed by the separate analysis code.

The protocols to be tested are specified in a config file, and have no imposed limits. As long as the correct binaries

This paper and associated code is available at https://github.com/wraithy/bigbinf
The OpenSSH-Portable installation used for HPN-SSH was compiled from https://github.com/rapier1/openssh-portable
The large hg38.fa.align.gz dataset used is available at http://hgdownload.cse.ucsc.edu/goldenPath/hg38/bigZips/
For more information, please see pubs.cs.uct.ac.za

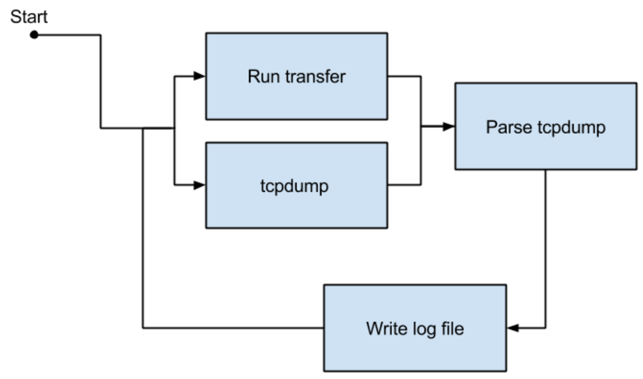


Figure 1: The sequence of actions performed by the testbed. This is repeated once for each protocol.

are installed on the machine, a user could test any transfer program as long as they know the command line arguments.

The testbed itself is designed to have minimal dependencies, while the optional analysis notebook makes heavy use of Python data science packages.

The logging mechanism makes use of the 'tcpdump' program [8], which comes with most Unix-like systems. As depicted in Figure 3, it watches a network interface (e.g. eth0) and logs information about packets which pass through. The program is run while each transfer is in progress, and the output is filtered to include only packets sent between Host 1 and Host 2. This output is used as the raw data for analysis.

The Python program for running the file transfers has been written to accept: the name of the network interface, the remote hostname (Host 2), the path of the file on Host 2, and a local path to copy the file to. It then resolves the IPs of each host, and for each protocol, runs a transfer in isolation.

As shown in Figure 1, it spawns a tcpdump subprocess which runs for precisely as long as the copy runs. The tcpdump program is started with filters, so that only traffic between the two hosts is captured. It then saves the output in a file.

This allows for a controlled environment, because tcpdump only captures while the copy is running, no other packets are included in the logs. Also, the copies are run programmatically and consecutively. Successive copies are not started until both the tcpdump and protocol processes have been closed, and the log file has been written. This means that they are all run in an identical (within reason) environment, but at the same time do not interfere with each other.

This test process is run multiple times for statistical reasons, generating multiple log files.

A Jupyter notebook [11] is provided to read in the dump files. This information can be aggregated, and then used to calculate metrics and display graphs. These include plots which are computed by looking at the time of each packet, and the size of its payload.

2.2 Test Case between UCT and UWC

The testbed is deployed between two hosts as a development environment for the testbed itself. Once ready, a full set of tests is run between them.

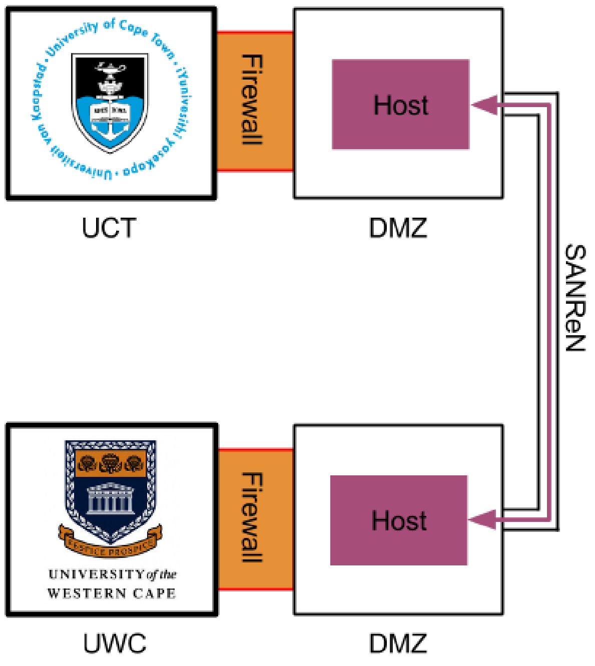


Figure 2: The hosts in their environment.

The hosts are virtual machines running at the South African National Bioinformatics Institute (UWC) and the Science DMZ (UCT). Both locations are close to the SANReN link, and outside institutional firewalls as depicted in Figure 2. This means that throttling is avoided, and ensures a minimum speed of 1Gbps.

Four different protocols were chosen to be tested. Two of the most widely used are FTP and SCP, and each of these has a more recent successor (GridFTP and HPN-SCP respectively).

Transfers using each of the four protocols are run while the network traffic is logged.

The testing environment is kept as stable as possible during tests, and multiple tests are run at different times of the day.

For each transfer, a copy is initiated from Host 1. A file from Host 2 is transferred to Host 1 using the particular protocol (see Figure 3).

The tests are run with 6 files. These are one small 5Mb file, and five files between 0.5Gb and 2.4Gb in 0.5Gb increments. These were all generated by reading chunks of a 2.4 GB gzipped sequence alignment file. Even though the protocols are agnostic of file format (they treat everything as binary), the sequence alignment file is a typical dataset that researchers would need to transfer.

2.3 Protocols

The OpenSSH [10] implementations of FTP (sftp) and SSH (scp) are tested. High Performance SSH (HPN-SSH) is a set of patches to OpenSSH which removes bottlenecks [12]. The scp binary from a patched, portable version of OpenSSH is tested for HPN. This is installed alongside the original so that both binaries are available.

The 'lite' version of GridFTP [1] is used. This means

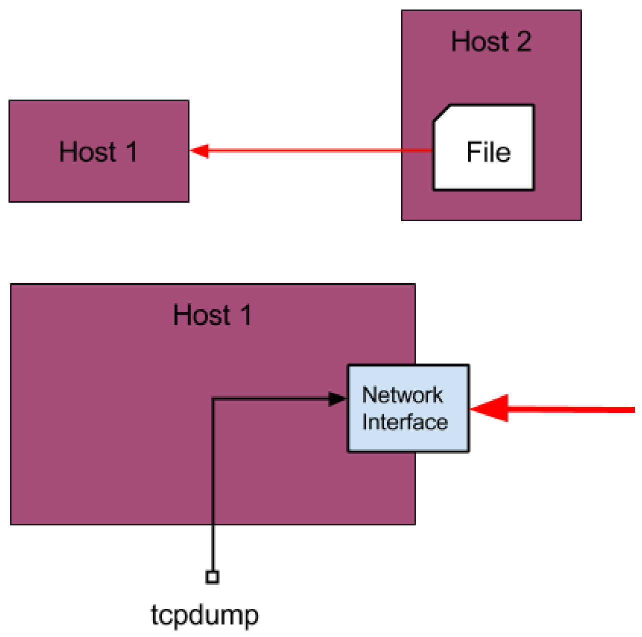


Figure 3: Host 1 copying a file and capturing packets.

that authentication is done via ssh as opposed a previously-configured certificate authority. This makes no difference to the file transfer itself, but it prevents unnecessary configuration of the testbed which can be quite complex in the case of 'full' GridFTP [7].

GridFTP supports data 'striping', which allows a transfer to be split across multiple hosts, potentially combining their bandwidths. This feature is not used, because the number of available nodes can vary greatly between configurations, and it would make comparisons with single-stream transfers unrealistic.

GridFTP also provides two modes of general operation. The default mode operates on top of the TCP stack, but there is a newer 'UDT' mode which is based on UDP. This mode aims to overcome congestion control bottlenecks found in TCP. Although Bresnahan et. al. found that the UDT mode outperformed the TCP mode [3], both modes are still tested. Note that despite it's name, the tcpdump program will capture UDP packets.

3. RESULTS

Results were collected by running the testbed at 13h00 and 3h00 each day, for a period of two weeks. For some of the tests, all the collected data (including information about each packet) was stored to disk. Because this is very space intensive (a single test run could generate multiple gigabytes of dump files), most of the tests only stored their aggregated data.

The measures were chosen by practicality. For example, behind the scenes information like window sizes is not collected, because the biggest concern for users is speed and size.

3.1 Network stability

Although each transfer in one test runs in the same environment, the time that tests are run can influence the results if the network is busier at certain times of the day.

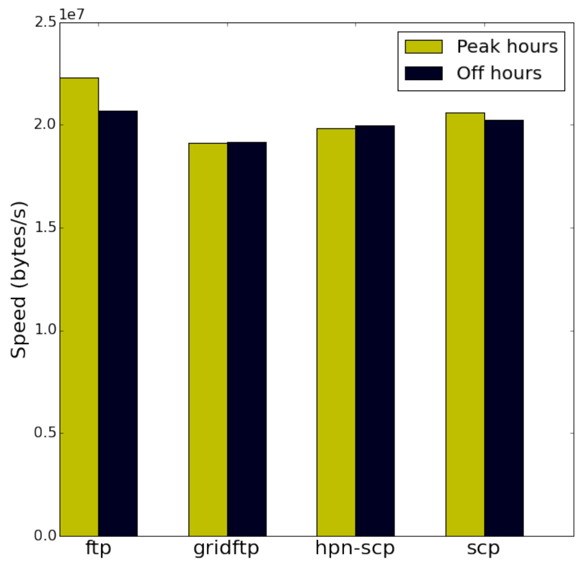


Figure 4: Comparison of speed during peak and off hours.

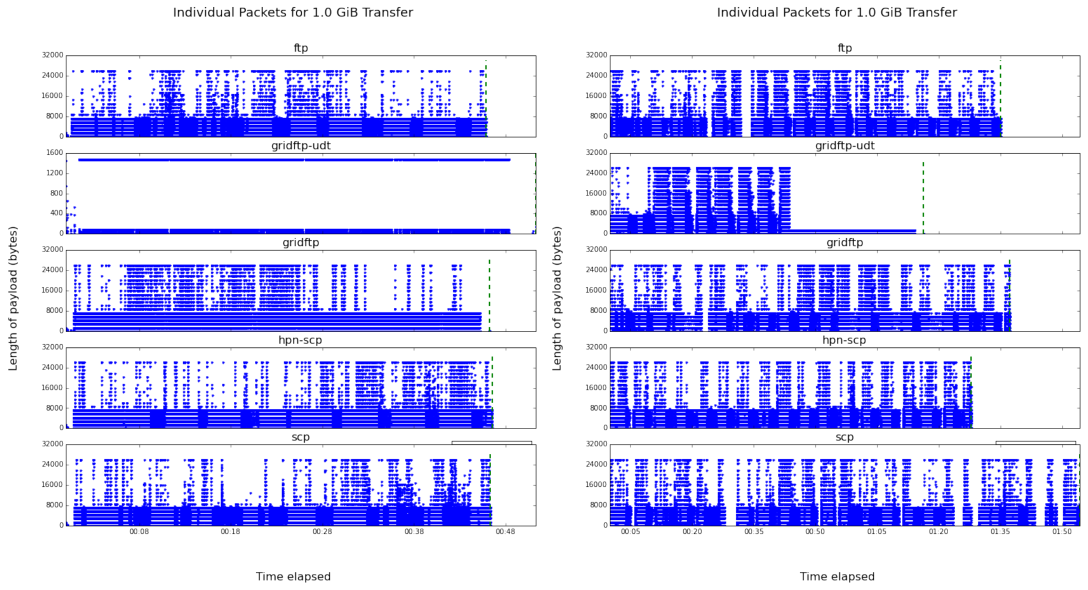
Protocol	p-value
ftp	0.5
hpn-scp	0.9
gridftp	1.0
scp	0.7

Table 1: p-values for a T-Test between the two sets, with null hypothesis that the speeds are the same.

Figure 4 is a conventional summary of the speed at various times in the day. "Peak hours" were considered to be between 11h00 and 15h00, and all tests were either in this time frame (peak) or not (off hours). Interestingly, the older protocols tend to be faster than their counterparts during peak hours, while the opposite is true during off hours. However, even though some might appear faster, a t-test between each set of subgroups reveals that the differences are not statistically significant. The p-values from this test can be seen in Table 1, and their values are clearly quite large (meaning that we cannot reject the null hypothesis).

While the stability of the network at each time affects the speed of the transfers, but it is sometimes useful to look other, more subtle effects as well. Figure 5 represents one complete transfer for each transfer protocol, for one file size (five rotations of Figure 1). This test is shown as a time series, where each point on the plot represents a packet. Both incoming and outgoing packets are plotted, and the height of each packet represents the size of it's payload. In this case, there was a confirmed instability at the specific date and time that Figure 5b was captured.

Firstly, comparing Figure 5a and Figure 5b confirms that transfers on a busy network do take longer. Secondly, the comparison shows how each transfer protocol handles the congestion. The fingerprints for the TCP based protocols are very similar, while the UDP based UDT mode of GridFTP changes in an obvious way. This mode tries to match it's packet sizes to the network MTU (Maximum Transmission



(a) Stable network (b) Unstable network

Figure 5: A graphical fingerprint of each transfer

Unit) for efficiency. The MTU for the ethernet interface on the test machine was 1500, and Figure 5a is an example of the transfer operating in ideal conditions, consistently achieving this size. However, it is clear from Figure 5b that the first half of the transfer was congested, and a bigger payload is forced into each packet. This is only one of the factors that influence transfer speed, but it clearly does have a negative impact.

3.2 Speed and Data Efficiency

Figure 6 is a comparison between all five protocols measuring the total speed, and total bytes. This is aggregated across all tests for each protocol. Note that the speed calculations only include downstream data, while the ‘transferred data’ measures include data that is sent from the client back to the server via control channels etc. The dashed red line in each plot represents the size of the transferred file, and can be used for quick reference to see if a particular transfer protocol sends more or less data than the filesize on average. If the point lies to the right of the red line, the transfer has an overhead on average. If the point is on the left, the protocol has managed to send compressed data in an efficient way.

The corresponding numerical data (including upstream packets) can be seen in Table 2.

This same data can be represented as a function of filesize. In this case, Figure 7b shows the transferred data as a ratio of the file’s size. A ratio of 100% would mean that the total number of bytes transferred is exactly the file size. Ratios larger and smaller than this are more and less data efficient respectively.

It is not obvious that any particular protocol uses better compression for this particular file, because data points lower than 100% are not consistent. It is obvious however, that GridFTP-UDT uses more data than the TCP based protocols.

In a practical sense, all 5 protocols have roughly the same efficiency, where compression is less than 0.6%, and overhead less than 5%.

Figure 7a is probably the most straightforward graph in this paper, as it compares the speed of each protocol. Surprisingly, the only protocols that are good consistently are ftp and scp, where the newer protocols do slightly worse for all file sizes. It must be noted that the differences in speed are quite small at this scale (the graph is scaled to 10^6). From a practical perspective, the slowest protocol only takes 7.64% longer than the fastest (for the 2.4Gb file size).

4. CONCLUSIONS

5. ACKNOWLEDGEMENTS

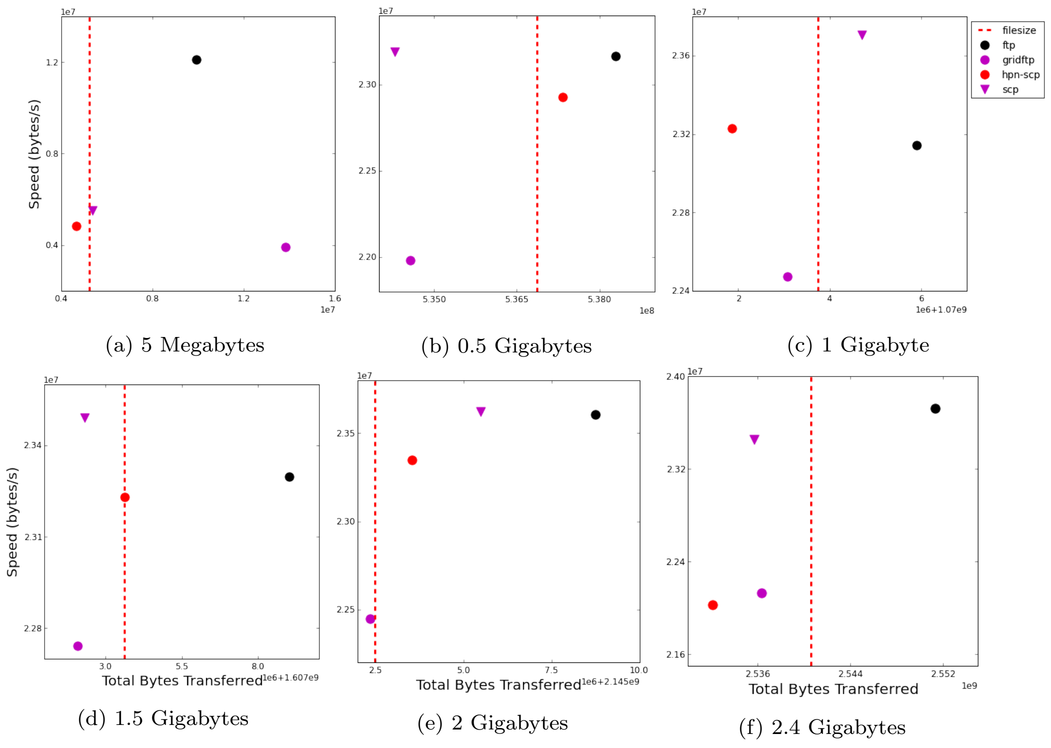
This project would not have been possible without access to virtual machines at both UCT and UWC. Mr. Peter van Heusden (South African Bioinformatics Institute) was extremely available during the project period, and kindly set up a VM in the SANBI DMZ. He also granted access to the Cisco switch which allowed rules for each protocol to be configured. Mr. Heine de Jager provided access to the UCT Science DMZ, and was very accommodating under strict security protocols. He was also extremely available, and helped fix multiple issues with the GridFTP setup and firewall rules.

6. REFERENCES

[1] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster. The globus striped gridftp framework and server. In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, page 54. IEEE Computer Society, 2005.

[2] M. Baker. Next-generation sequencing: adjusting to data overload. *nature methods*, 7(7):495–499, 2010.

[3] J. Bresnahan, M. Link, R. Kettimuthu, and I. Foster. Udt as an alternative transport protocol for gridftp. In



(a) 5 Megabytes (b) 0.5 Gigabytes (c) 1 Gigabyte

(d) 1.5 Gigabytes (e) 2 Gigabytes (f) 2.4 Gigabytes

Figure 6: Aggregated data per protocol

International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT), pages 21–22. Citeseer, 2009.

[4] P. J. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic acids research*, 38(6):1767–1771, 2010.

[5] S. Deorowicz and S. Grabowski. Compression of dna sequence reads in fastq format. *Bioinformatics*, 27(6):860–862, 2011.

[6] X. Fan and M. Munson. Petabytes in motion: Ultra high speed transport of media files a theoretical study and its engineering practice of aspera faspâĎć over 10gbps wans with leading storage systems. In *SMPTE Conferences*, volume 2010, pages 2–13. Society of Motion Picture and Television Engineers, 2010.

[7] Globus. The south african national research network. http://toolkit.globus.org/toolkit/data/gridftp/quickstart.html, 2015. Accessed: 2015-09-06.

[8] V. Jacobson, C. Leres, and S. McCanne. Tcpdump. http://www.tcpdump.org/, 2015. Accessed: 2015-10-18.

[9] V. Marx. Biology: The big challenges of big data. *Nature*, 498(7453):255–260, 2013.

[10] OpenBSD. Openssh. http://www.openssh.com/, 2015. Accessed: 2015-09-11.

[11] M. Ragan-Kelley, F. Perez, B. Granger, T. Kluyver, P. Ivanov, J. Frederic, and M. Bussonier. The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication. *AGU Fall Meeting Abstracts*, page D7, Dec. 2014.

[12] C. Rapier and B. Bennett. High speed bulk data transfer using the ssh protocol. In *Proceedings of the 15th ACM Mardi Gras conference: From lightweight mash-ups to lambda grids: Understanding the spectrum of distributed computing requirements, applications, tools, infrastructures, interoperability, and the incremental adoption of key capabilities*, page 11. ACM, 2008.

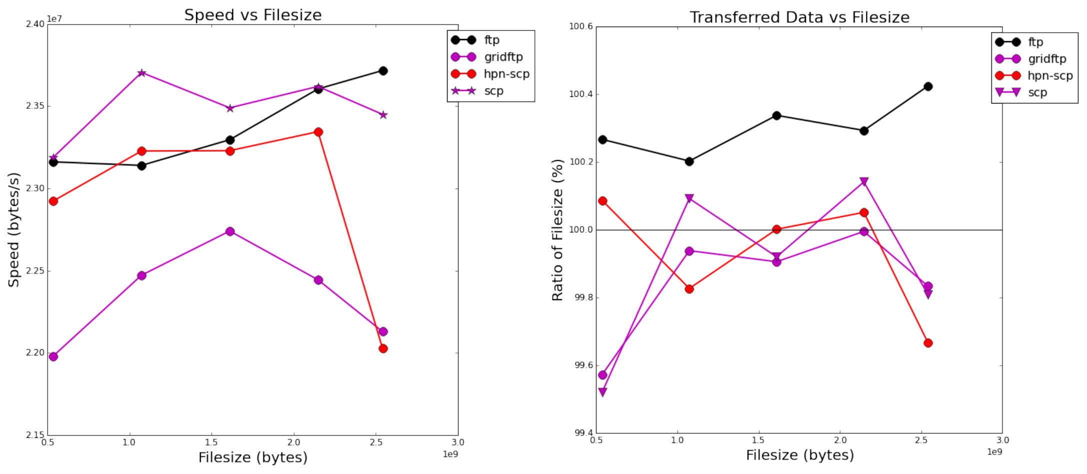
[13] SAMTools. Sequence alignment/map format specification. https://samtools.github.io/hts-specs/SAMv1.pdf, 2015. Accessed: 2015-04-27.

[14] SANReN. The south african national research network. http://www.sanren.ac.za/, 2015. Accessed: 2015-09-01.

[15] J. Shendure and H. Ji. Next-generation dna sequencing. *Nature biotechnology*, 26(10):1135–1145, 2008.

File Size (bytes)	Protocol	Bytes Down	Bytes Up	Bytes Total	Ratio (%)	Time (s)	Speed (bytes/s)
5242880	ftp	9922928.4	14540.1	9937468.5	189.5	1.4	12094979.9
	gridftp	5080052.9	8774708.2	13854761.0	264.3	1.3	3886209.6
	hpn-scp	4666512.3	6496.9	4673009.1	89.1	1.0	4829022.0
	scp	5224851.0	160461.1	5385312.1	102.7	1.0	5506684.1
536870912	ftp	537007757.3	1291980.4	538299737.7	100.3	23.3	23161405.3
	gridftp	534573783.6	4429.3	534578212.9	99.6	24.5	21980615.4
	hpn-scp	537171422.0	166915.8	537338337.8	100.1	23.6	22923730.8
	scp	534104526.3	188454.4	534292980.7	99.5	23.1	23189047.2
1073741824	ftp	1073357571.6	2553835.6	1075911407.2	100.2	46.6	23139968.4
	gridftp	1073079145.8	4428.0	1073083573.8	99.9	47.9	22472118.5
	hpn-scp	1071535742.8	342785.1	1071878527.9	99.8	46.2	23227507.5
	scp	1074336336.4	382952.0	1074719288.4	100.1	45.4	23704931.9
1610612736	ftp	1612216037.3	3828633.2	1616044670.5	100.3	69.3	23295999.1
	gridftp	1609088422.9	4426.1	1609092849.0	99.9	70.9	22740693.2
	hpn-scp	1610127154.9	511912.7	1610639067.7	100.0	69.4	23229536.9
	scp	1608744654.9	577390.4	1609322045.4	99.9	68.5	23490043.3
2147479552	ftp	2148673854.9	5075559.6	2153749414.5	100.3	91.1	23604911.2
	gridftp	2147364435.1	4429.3	2147368864.3	100.0	95.7	22444177.1
	hpn-scp	2147870110.0	694075.5	2148564185.5	100.1	92.1	23345011.5
	scp	2149719545.9	774732.1	2150494278.1	100.1	91.1	23620164.9
2540610608	ftp	2545393837.5	5996149.6	2551389987.0	100.4	107.4	23717061.8
	gridftp	2536383335.4	4406.8	2536387742.2	99.8	114.7	22128903.4
	hpn-scp	2531341159.6	792915.0	2532134074.6	99.7	115.6	22026116.8
	scp	2534857393.3	913497.4	2535770890.7	99.8	108.2	23450202.2

Table 2: Numerical data aggregated data per protocol



(a) Speed

(b) Data Efficiency

Figure 7: Metrics calculated per file size