The Widening Gulf between Genomics Data Generation and Consumption: A Practical Guide to Big Data Transfer Technology

Frank A. Feltus[1], Joseph R. Breen III[2], Juan Deng[3], Ryan S. Izard[3], Christopher A. Konger[4], Walter B. Ligon III[3], Don Preuss[5] and Kuang-Ching Wang[3]

[1]Department of Genetics and Biochemistry, Clemson University, Clemson, SC, USA. [2]University of Utah Center for High Performance Computing, Salt Lake City, UT, USA. [3]Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, USA.[4]Clemson Computing and Information Technology, Clemson University, Anderson, SC, USA. [5]NIH/NLM/NCBI, Bethesda, MD, USA.

Supplementary Issue: Current Developments in RNA Sequence Analysis

ABSTRACT: In the last decade, high-throughput DNA sequencing has become a disruptive technology and pushed the life sciences into a distributed ecosystem of sequence data producers and consumers. Given the power of genomics and declining sequencing costs, biology is an emerging "Big Data" discipline that will soon enter the exabyte data range when all subdisciplines are combined. These datasets must be transferred across commercial and research networks in creative ways since sending data without thought can have serious consequences on data processing time frames. Thus, it is imperative that biologists, bioinformaticians, and information technology engineers recalibrate data processing paradigms to fit this emerging reality. This review attempts to provide a snapshot of Big Data transfer across networks, which is often overlooked by many biologists. Specifically, we discuss four key areas: 1) data transfer networks, protocols, and applications; 2) data transfer security including encryption, access, firewalls, and the Science DMZ; 3) data flow control with software-defined networking; and 4) data storage, staging, archiving and access. A primary intention of this article is to orient the biologist in key aspects of the data transfer process in order to frame their genomics-oriented needs to enterprise IT professionals.

KEYWORDS: data life cycle, Big Data, networks, software-defined networking

SUPPLEMENT: Current Developments in RNA Sequence Analysis

CITATION: Feltus et al. The Widening Gulf between Genomics Data Generation and Consumption: A Practical Guide to Big Data Transfer Technology. *Bioinformatics and Biology Insights* 2015:9(S1) 9–19 doi: 10.4137/BBI.S28988.

TYPE: Review

RECEIVED: June 09, 2015. **RESUBMITTED:** August 10, 2015. **ACCEPTED FOR PUBLICATION:** August 12, 2015.

ACADEMIC EDITOR: JT Efird, Associate Editor

PEER REVIEW: Six peer reviewers contributed to the peer review report. Reviewers' reports totaled 1968 words, excluding any confidential comments to the academic editor.

FUNDING: We would like to thank the National Science Foundation for funding of our research efforts through these awards: #58501934 (JRB), #1245936 (KCW, FAF), #1447771 (WBL, FAF), and #1443040 (FAF, KCW). This research was supported in part by the Intramural Research Program of the NIH National Library of Medicine (DP). The authors confirm that the funder had no influence over the study design, content of the article, or selection of this journal.

COMPETING INTERESTS: Authors disclose no potential conflicts of interest.

CORRESPONDENCE: FFELTUS@clemson.edu

Paper subject to independent expert blind peer review. All editorial decisions made by independent academic editor. Upon submission manuscript was subject to anti-plagiarism scanning. Prior to publication all authors have given signed confirmation of agreement to article publication and compliance with all applicable ethical and legal requirements, including the accuracy of author and contributor information, disclosure of competing interests and funding sources, compliance with ethical requirements relating to human and animal study participants, and compliance with any copyright requirements of third parties. This journal is a member of the Committee on Publication Ethics (COPE).

Published by Libertas Academica. Learn more about this journal.

Genomics is Big Data

The phrase "Big Data" is being used across a diverse array of disciplines, each with its own interpretation of its meaning. To the 8,000 particle physicist consumers of Large Hadron Collider measurements,[1] Big Data means having access to a constant stream of over 42 petabytes/year of particle collision sensor data through a three-tier network of high-performance computing (HPC) data centers across the world. Much of the heavy lifting of raw data preprocessing occurs within the massive distributed software and hardware infrastructure of network, storage, and compute systems, allowing the researcher to mine collision events in Big Data to make amazing discoveries like the observation of the Higgs boson.[2]

To the consumer of genomics data, who are now resident in all life science disciplines ranging from ecologist to oncologist, Big Data might mean a dozen text files, each derived from a raw image file produced by the DNA sequencer, filled with tens of millions of DNA sequence strings summing to tens of gigabytes of data. That researcher might have trouble transferring the data from a DNA sequencing instrument to the computational workflow site, and may even resort to mailing a hard drive to a collaborator if the network bandwidth is too limited. Once data reach the target file system, the researcher must balance finite storage and processing resources to process raw DNA sequence data into powerful constructs like a new genome (eg, coffee[3]) or a biomarker signature of cancer drug resistance.[4] To these researchers, modern DNA sequence analysis is a Big Data endeavor. If all the DNA sequencing instruments were stacked next to the LHC collider, anchoring only four, albeit much larger, instruments, and the annual DNA output is summed, even the particle physicist would probably agree that genomics is a now a Big Data discipline.

While rapid DNA sequencing has been around for 40 years,[5] only recently did it enter the Big Data zone. Many

biologists use deep DNA sequencing technology, which generates gigabyte dataset files, as the engine for a growing list of powerful applications. High-throughput sequencing allows the rapid construction of entire DNA molecules (reference chromosome assemblies) or partial sequencing of DNA/RNA molecules at deep enough sampling for gene expression and polymorphism discoveries. These data troves are stored in burgeoning data repositories such as the Sequence Read Archive (SRA) that can be accessed National Center for Biotechnology Information (NCBI), the European Bioinformatics Institute (EBI), and the DNA Data Bank of Japan (DDBJ).[6] SRA contains 4.0 petabytes of data deposited in the last 6 years with geometric growth,[7] CGHub,[8] which stores National Cancer Institute data from petabyte scale projects including The Cancer Genomics Atlas[9] and Illumina's BaseSpace.[10] Data access at these repositories has inspired or taken advantage of advanced data transfer networks such as Internet2 100Gbit Advanced Layer 2 Services (AL2S[11]) and GENI,[12] data flow control applications such as Globus Online[13] and Aspera,[14] and cyberinfrastructure environments (reviewed in[15]) including XSEDE,[16] iPlant,[17] Open Science Grid,[18] ELIXIR,[19] UPPNEX,[20] CloudBioLinux,[21] BGI-Cloud,[22] and CloudLab.[23] Diving sequencing costs and the power of the high-throughput sequencing measurement paradigm will surely accelerate genomics data accumulation and processing demands.

No matter which discipline generates Big Data, it must be transferred, stored, processed, and archived for reproduction of results as well as unforeseen data mining opportunities. The high-throughput DNA sequence data life cycle is shown in Figure 1. Note that this life cycle includes data genesis at the DNA sequencer and includes multiple networks data must flow through, both between organizations and within an HPC system, where workflow software crunches data flowing through networks within the data center. Most researchers are focused on biological experimental design, yet Big Data can easily overwhelm the average researcher's compute and storage resource allocations. Therefore, careful consideration of the data life cycle must be managed by the researcher or abstracted through automated tools. In essence, Big Data biology is still driven by the tried and true scientific method of hypothesis testing through experimentation, but Big Data forces a concomitant "technology experimental design" that can be complex and often beyond the capacity of both the experimentalist and standard IT support within a research enterprise.

DNA sequencers and other instruments inject measurements of the natural world into the biological data ecosystem. While data are created with a specific purpose in mind, one can view genomic data generators as connections to the larger world of scientific collaboration. This collaboration occurs through networks like the Internet and Internet2, where the data flows between the original investigators' compute resources and enters repositories such as NCBI where creative data recombination and analysis can produce evidence for unforeseen hypotheses. This collaborative work and data flow allows the original and

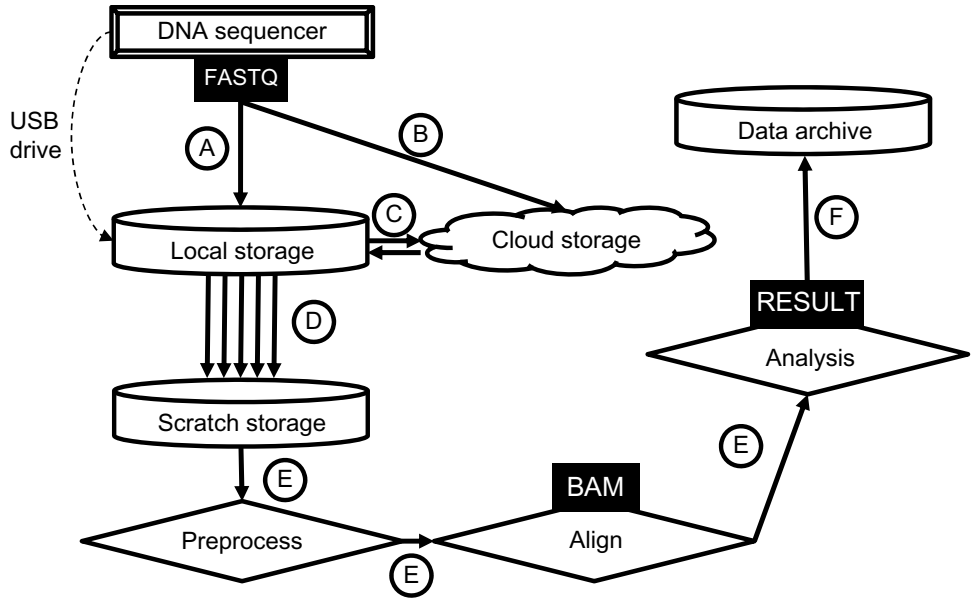

Figure 1. High-throughput DNA sequence data flows across multiple networks. A representative data analysis lifecycle begins when physical molecules (DNA, RNA) from biological samples are measured at a sequencing facility. DNA text data files (FASTQ) are sent to the researcher across inter-organization networks A or B to a local or cloud storage site. Sometimes researchers receive datasets via mailed hard drives across the postal network (see note below for clarification). DNA sequences are then transferred to the workflow compute site (eg, /scratch) across network D, which could be another organization or within the user's HPC compute system. Multiple arrows are shown for network D, as one or more data streams are possible during transfer. Once data is transferred to HPC nodes, workflows including preprocess, genome alignment, and downstream analyses are subject to internal network E constraints (eg, Infiniband). Final data results (including raw FASTQ) are then transferred over network F into a data archive. All hardware has constraints. Each network might be capable of different transfer rates (1M> 100 G bits per second). Each file system has finite storage, variable read/write rates, variable numbers of controllers, and parallelization. Note that only one flow is shown, yet compute could occur across multiple systems.

opportunistic investigators to test hypotheses far beyond the original context of the data. Given the potential of Big Data mining, it is essential for twenty-first century biologists to be aware of the underlying data transfer network technology.

Data transfer requests are affected by every link along the way, including the source computer (and storage), the remote computer (and storage), and all of the intermediate network connections. This article is designed to raise awareness of these myriad issues and discusses potential solutions to the travails one is likely to encounter. Specifically, we will describe the current state of the hardware technology that links all stages of the genomics data life cycle. We will describe the current limits of network, storage, and data security and access technology. A primary intention of this article is to orient the genomics researcher in key aspects of the data transfer processes in order to design networked analysis systems for a discrete research group, and to help the experimentalist frame his/her own genomics-oriented needs to enterprise IT professionals.

Data Transfer Networks

Data transfer networks and new network technologies are great enablers of genomic data lifecycles. Rapidly increasing network speeds (eg, 100 Gbit/s Internet2 AL2S) and computer power hold the promise of moving datasets from instrument to compute and finally to end result in time frames that might someday be useful for near-real-time feedback. Modern technologies such as software-defined networking (SDN) enable virtualization of the network, the ability to optimize paths, schedule transfer jobs, and the ability to contain data flows with specific security characteristics. However, as with most technologies, scientists will go through much iteration in order to make these technologies enable their discovery programs and not just be "cool technology" on the horizon. Scientists who need to deploy advanced networks on their systems should understand what is possible now, set strong development goals, and test these networks with real-world datasets. For optimum experience, scientists should build strong collaborations in their IT departments with progressive network administrators, system administrators, storage administrators, and development staff in order to truly strive toward their specific goals. Often, IT professionals are focused on the construction of robust general-purpose "production" systems, and the scientist may need to explicitly communicate the emerging data needs for genomics applications.

Network data flow from source to destination is often conceptualized as the seven-layer open systems Interconnection (OSI) model (Fig. 2). Modern genomics data lifecycles require modern network hardware that can sustain network flows on the order of 1 Gbit/s, or preferably greater, across these layers. To put into perspective, a 1 Gbit/s data flow transfers a terabyte of data in approximately three hours. If a network can sustain a 10 Gbit/s data flow, the network can transfer the same terabyte of data in 20 minutes.[24] As networks improve (ie, 40 and 100 Gbit/s with 400–1,000 Gbit/s on the horizon), these transfer times drop dramatically and a scientist is able to use the network to transfer larger datasets faster.

Layer	Data unit	Function	Examples
7. Application	Data	High-level APIs, including resource sharing, remote file access, directory services and virtual terminals	HTTP, FTP, SMTP
6. Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption	ASCII, EBCDIC, JPEG
5. Session		Managing communication sessions, ie continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes	RPC, PAP
4. Transport	Segments	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing	TCP, UDP
3. Network	Packet/Datagram	Structuring and managing a multi-node network, including addressing, routing and traffic control	IPv4, IPv6, IPsec, AppleTalk
2. Data link	Bit/Frame	Reliable transmission of data frames between two nodes connected by a physical layer	IEEE 802.3/802.2
1. Physical	Bit	Transmission and reception of raw bit streams over a physical medium	Fiber, Copper twisted wires

Figure 2. Seven-layer open systems interconnection (OSI) model. This model of data communication abstraction layers allows for the conceptualization of data transfer networks. Adapted from http://en.wikipedia.org/wiki/OSI_model under terms of the Creative Commons Attribution-Share-Alike License 3.0.

However, maximum data flow rates are often measured under ideal conditions since many variables affect the actual transfers, such as network interface card (NIC) capability, intermediate processing nodes, firewalls, routers, network congestion, computer bus speed, file system speed, and other factors.

The ability to sustain a flow (or collection of flows) at 1 or 10 Gbit/s transfer speeds is a function of maintaining an end-to-end path that can sustain the transfer speed. In simple terms, the end-to-end transfer rates are limited by the slowest link, wherever it might exist within a path. Typical desktop hosts have 1 Gbit/s NIC connections, while modern servers holding large datasets have 10 Gbit/s or better NICs. However, NICs in servers and DNA sequencers can have widely varying throughputs. Across a data transfer path, the speed of the slowest NIC defines the theoretical transfer rate across that path.

Unfortunately, finding the slowest NIC is not going to define the actual maximum data transfer rate across a network path. NICs plug into a computer bus, such as the PCI Express bus, which can have different speeds (ie, 2.5–128 Gbit/s) depending on the version and number of lanes supported. NICs also have varying amounts of data buffer memory and internal architectures, some of which might support off-loading of specific network features to hardware for acceleration [eg, network processing unit (NPU)]. The actual network devices (routers and switches) have similar limitations. Routers and switches take the data from the computer and pass it across the local area network (LAN) and the wide area network (WAN). Each point-to-point set of interfaces through which the data passes can affect the data speeds. Each point-to-point set of interfaces has a particular speed, an existing amount of traffic, a set of memory buffers, and an existing amount of loss or errors.

The best way to see how well a network is functioning is to test it. Network engineers have many tools at their disposal such as perfSONAR.[25] It should be noted that, often, domain-specific engineers typically focus on their specific subsystems including networking, storage, and security, so a data source to destination test by the data consumer will provide a realistic measure of network performance. This is a good point to remind the reader that networks are shared resources, so data flows from other users (network congestion) will obviously slow down the expected data transfer rates and tests should be spread out at different times. Realistic transfer measurements, coupled with knowledge of the slowest hardware point on the path, provide a data flow baseline. If the baseline is near the expected transfer maxima, then all is well. However, if performance is significantly less than expected, it may be time for a discussion between the data consumer and IT engineers. Often, there is a malfunctioning hardware component or misconfigured device that can be rectified. If the data pipe is running optimally and performance is still not what the researcher needs, then it may be possible to relieve bottlenecks (eg, see storage section below) through focused hardware investments and possibly through changing I/O allocation policies for heavy data transfer users.

Data Transfer Network Protocols

In addition to physical hardware, network protocols, transfer methods, and parameters used in transfers can all affect the ability of a scientist to fully realize the goals of migrating data at full line rate from source to destination endpoint. Network protocols define how data passes over the network in a manner in which an endpoint can decode in a decipherable way. The efficient transfer of large datasets typically involves the creation of "chunks" that an application sends across the various links to the destination. The loss of a particular chunk can be highly impactful. Such losses can happen for a variety of reasons, including exhaustion of packet buffers in the network equipment, reaching the peak capacity for a specific link, failing or intermittent optics, or even degradation of fiber or copper infrastructure. Applications can rely on network protocols to handle resending data, or the applications can handle the resending by themselves.

Applications primarily use two specific Layer 4 transport network protocols for transferring data: transmission control protocol (TCP) and user datagram protocol (UDP). Both these transport protocols layer on top of the Layer 3 Internet Protocol (IP). An application using TCP relies on the transport protocol to send chunks of data, each of which receive an acknowledgment when the destination receives them. There is a maximum amount of unacknowledged data that can be in transit on the links (the "TCP Window"), which means that the sender must pause when that limit is reached. Likewise, either side may detect a lost packet, and then retransmitting of the missing chunks proceeds. For high performance, such losses must be minimal (ideally zero), since a loss of only 1 packet in 22,000 can cause a significant drop in performance.[26] An application using UDP allows the transport protocol to send data in an "unreliable" or "best-effort" method. The protocol continues to send data as rapidly as possible without tracking and resending missing chunks. Since the transport protocol does not handle the tracking of the chunks, the applications running on both ends must coordinate those activities. Both UDP and TCP protocols work, but as genomics data sizes increase, the way these protocols are tuned and the raw network fidelity can have a huge impact transfer rates, especially over long distances.

Data Transfer Applications

In order for the scientist to move datasets over the network, he or she interacts with a software application to handle the transfers. These Layer 7 applications, regardless of the underlying protocols, move the data in either a serial fashion (ie, one file after another) or a parallel fashion (ie, two or more data streams at the same time). Both methods can effectively fill an end-to-end network path to capacity, but parallel transfers can often radically speed up transfer. While any Layer 7 transfer data application can transfer data in parallel streams by piping files in byte offsets and reconstructing them, several software packages explicitly use parallelization in their design including Aspera fasp[14] and Globus Online's gridFTP technology.[27] Additional variables such as file size, number of files, encryption

requirements (see next section), the manner that the application interacts with the network, and storage considerations all affect the actual performance of the transfers. Generally, compressing lots of little files into several larger files and using a parallel transfer application yields the best results with typical datasets.

Data Transfer Security

Prior to the transfer of genomics data to a new location, a scientist must consider whether regulations or common sense require data encryption, stringent access control, and firewalls. For small datasets, there is little impact on encryption and decryption, but for large genomics datasets these procedures can delay data entry into post-transfer analysis workflows. The scientist must also understand whether these encryption requirements apply to the data while it resides on disk, while it is "in flight" between two endpoints, or both. Without file or transfer encryption, a malicious eavesdropper can tap a wire or switch and read all the contents of the file in plain format. With file encryption but no transfer encryption, the eavesdropper can capture the encrypted files and apply decryption techniques to the full files. With both file encryption and transfer encryption, the eavesdropper can capture encrypted segments of transfers, but he or she must decrypt how the files exist on the network and then what the files actually are. For the purposes of transfer, the act of encrypting may incur a performance penalty on the transfer itself.

The three basic principles of information security, generally known as CIA, are *confidentiality* (no unauthorized disclosure of data), *integrity* (no unauthorized modification of data), and *availability* (ensuring the accessibility of data to legitimate users). Different security mechanisms provide different degrees of confidentiality, integrity, and availability. A data center generally relies on access control and other security practices to achieve the CIA. Access control is a security mechanism that protects data from unauthenticated users and determines the level of authorization for an authenticated user.

Access control provides the functions of *authentication* and *authorization*. Authentication identifies a legitimate user of the protected data. The proof of identity can be one or a mix of the following: 1) what the user knows (eg, password, PIN, pass phrase, or challenge response where the user must answer a question); 2) what the user has (eg, security token, ID card), 3) what the user is or does (eg, finger print, retinal pattern, signature, voice, face). Authorization grants or denies permission to an authenticated user to perform certain operations on protected data based on data security policies. Access control also traces and logs the operations performed by a user for auditing purpose.

When a user requests to perform certain action (eg, data download, data modification, data deletion) on the data on the servers, the user submits his/her identity and credential (eg, username and password), which together uniquely identify the user. The access control system verifies the identity and credential against its identity database. If the authentication is successful, then the access control system examines its data security policy database, which stores policies defining the authorizations for authenticated users. For example, an authenticated user may be authorized to download data but not to change data. If the action that the user requests to perform on the data is authorized according to the data policies, then the access control system will grant the permission to the user. Otherwise, even though a user is successfully authenticated, the action request will be denied. Access control enforces security controls on data and the servers storing data.

In addition to data encryption and access, networks are protected by firewalls. A firewall is a network security system that protects the infrastructure, service, and data of a network by enforcing controls on the incoming and outgoing network traffic at the network border. Every packet passing the border is examined against a set of firewall rules preinstalled in the firewall. Firewall provides security at the cost of introducing traffic delays, as each packet is held at the firewall for screening. For most of the common missions and services today running in a campus network, the impacts of traffic delays caused by firewalls are small and tolerable, considering the security benefits they offer. However, in large dataset transfer scenarios, firewalls can introduce latencies, bandwidth ceilings, and, occasionally, dropped packets if under heavy load. Firewalls serve a necessary purpose, but scientists and IT engineers have to be aware of the workflow and the impacts to the specific workflows in order to achieve proper balance of security and performance.

Campus networks are general-purpose networks, which are not designed and optimized for data-intensive science applications. Scientific data flows tend to be large and long-lived, known as "elephant flows", whereas most campus use is small and short-lived (eg, web traffic) and known as "mouse flows". It is recognized that stateful firewalls have become performance bottlenecks for data-intensive applications.[28] The Science DMZ model (Fig. 3) was proposed to solve the problem by removing the performance limits caused by a stateful firewall.[29] A Science DMZ is a subnetwork of a campus network that is structured to be safe but without the performance limits that result from a stateful firewall. The idea behind the Science DMZ is to create a "demilitarized zone" that is as close as possible to the network edge, outside the network enterprise, and put the special switches and gears for data-intensive systems in that zone. By bringing the data-intensive systems together and close to the network edge, the path to reach these systems is shortened, thereby reducing the communication time. Since Science DMZ is placed outside of the network enterprise, traffic to the Science DMZ does not go through the firewall, hence removing the firewall performance bottleneck. Network gear in the Science DMZ can be optimized for high-performance (100 Gbit/s interfaces) and large buffers, without requiring the entire enterprise to support these more costly solutions. Without firewall protection, there must be a substitute security mechanism to protect the Science DMZ. The substitute security mechanism of the Science DMZ might be an alternative such as setting access

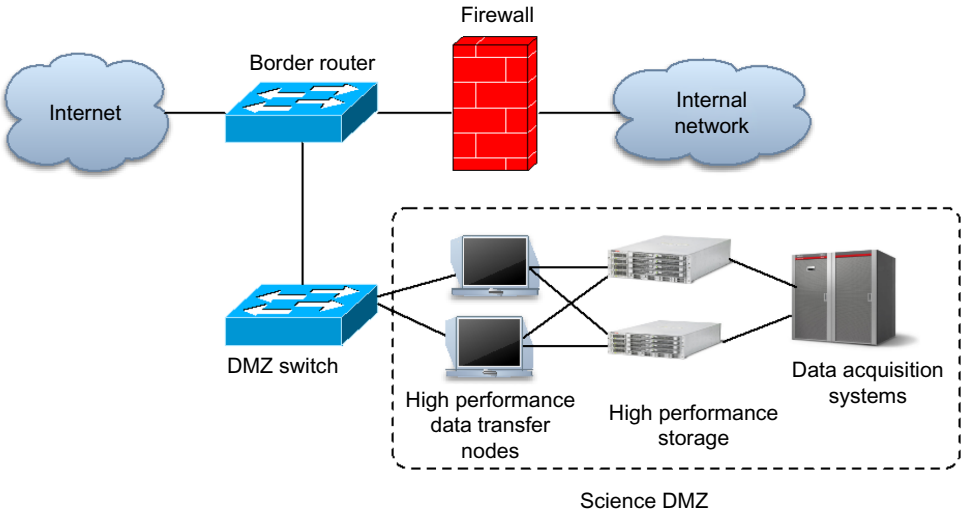

Figure 3. Science DMZ architecture. Network traffic is typically routed through a firewall prior to entering an organization's internal network. Packet inspection causes friction and a reduction in flow rate. Science DMZs are configured so that data flow bypasses the firewall using a dedicated DMZ switch.

control lists (ACLs) in the DMZ router, manipulating route tables, utilizing devices that manipulate data plane flows, etc. These security approaches allow traffic from particular known sources to specific known destinations and ports. These methods effectively provide functionality approaching key aspects of the enterprise firewall and, simultaneously, allow the full throughput of data.

Data Flow Control with Software-defined Networking

SDN is an important advance toward allowing more effective scheduling of I/O resources used in WAN transfers. SDN is a new network paradigm that enables applications and network operators great freedom to programmatically customize their networks. This paradigm accomplishes this customization by separating the *control plane* that decides how to forward traffic from the *data plane* that performs actual traffic forwarding, as seen in Figure 4A. Network devices, hence, become simple, pure packet-forwarding devices. An application programming interface (API) is provided for network operators and applications, which allows them to enforce control over packet forwarding rules on network devices.

Why SDN? In traditional non-SDN networks, a flow of packets traversing the hops between communication ends are handled at network devices (routers and switches) according to well-defined, standard protocols. Neither the application initiating the flow nor the network operator has full control over the flow. The complex and static nature of traditional networks makes them difficult to meet the requirements of various stakeholders and applications. Network operators face the challenges

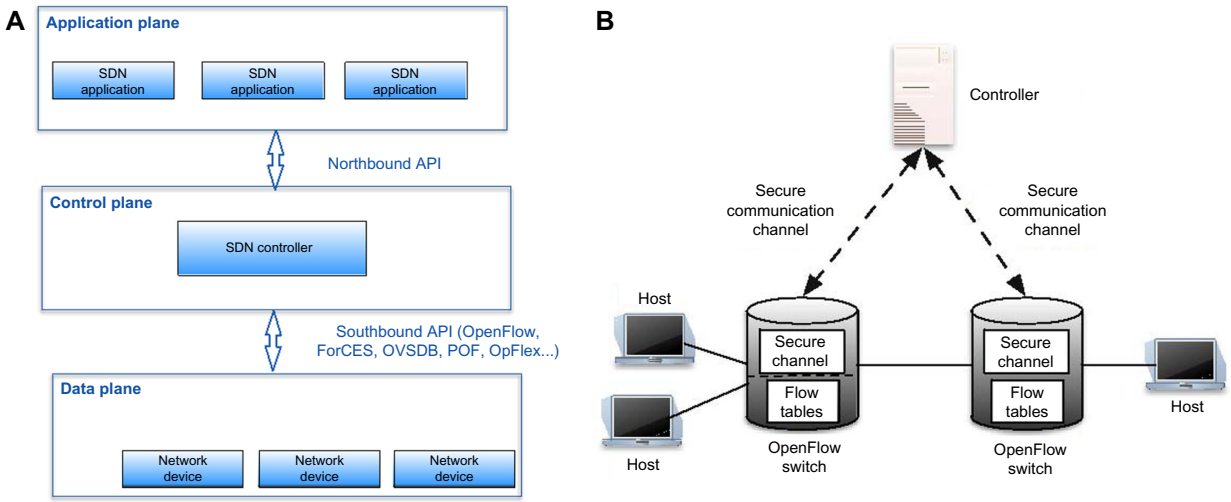

Figure 4. Data transfer control via software-defined networking (SDN). (**A**) The general model of SDN is shown where the control plane that decides how to forward traffic is separated from the data plane that performs actual traffic forwarding. (**B**) OpenFlow SDN architecture. The controller is software application that programs the flow tables of each OpenFlow switch, using the OpenFlow protocol.

of difficult management tasks.[30] To enforce network policies, network operators need to manually configure each individual network device from various vendors using vendor-specific configuration tools and interfaces, which can take a long time. For network researchers desiring to experiment with new network protocols, there is almost no practical way to conduct the experiment on traditional non-SDN networks.[31] SDN is a mechanism to return data flow control back to the data transfer agents.

Genomics scientists are experiencing increasing needs for transferring Big Data across long-distance geographic locations. Big Data transfer applications have demanding, multi-domain bandwidth requirements. While upgrading existing networks with larger pipes and faster switches significantly increase end-to-end bandwidth, there is a greater need for simple and scalable end-to-end network architectures and implementations that can most efficiently utilize available network bandwidth.[28] Existing solutions in traditional non-SDN build bandwidth-guaranteed virtual circuits between the communication ends. A virtual circuit is a way of transferring data over packet-switched networks in such a way that it appears as though there is a dedicated physical layer link between the source and destination for the data transfer. While these solutions are widely adopted, several challenges remain[28]:

- Campus networks or enterprise networks were not architected, deployed, or optimized for Big Data transfers. Tailoring networks to better meet the needs of Big Data transfers is nontrivial with traditional networks.
- Bandwidth-guaranteed virtual circuits need to be manually built, configured, and managed by network operators between two communication points each time a new data transfer starts and need to be torn down when the transfer ends in order to release network resource. For a data-transfer-intensive environment, an automatic mechanism is called for.

SDN, emerging as a new network paradigm, offers the capabilities that can be leveraged to build solutions for Big Data transfer that address the challenges. Various SDN-based solutions for Big Data transfers have been proposed[28,32–34] OpenFlow is one embodiment of SDN concept and experiences high popularity.[31] The architecture of OpenFlow is depicted in Figure 4B. A centralized controller controls OpenFlow switches. An OpenFlow Switch has one or more flow tables. A flow table consists of multiple flow entries. Each flow entry has match fields, counters, and instructions. Data packaged in frames arriving at an OpenFlow switch are compared with the match fields of each entry, and if there is a match, the frame is processed according to the instructions in that entry. Counters are used to keep statistics about the frames. If no match is found, the OpenFlow switch encapsulates the frame and sends it the controller. The controller is software application that programs the flow tables of each OpenFlow switch, using the OpenFlow protocol. The communication between the controller and an OpenFlow switch is through a secure communication channel.

In the context of big genomics data, SDN technology coupled with innovative network approaches can decrease the time to transfer a set of large DNA sequence files or other type of biological data from one remote repository to a researcher for timely analysis. Steroid OpenFlow Service (SOS[33]) is an SDN-based solution designed to seamlessly increase the throughput of large data transfers across large networks (with wide geographical distances and latencies which increase the delay–bandwidth product). A diagram depicting the SOS architecture is shown in Figure 5. SOS leverages the fact that TCP is typically unable to utilize all the bandwidth available over large networks. TCP is only permitted to send its window-size number of packets before receiving an acknowledgment. In a large network, this means the sender must wait idle for the acknowledgment prior to sending additional data packets. SOS works by redirecting the designated TCP connection of such long-distance data transfer to a local SOS agent via an in-path OpenFlow switch. The local SOS agent transparently acknowledges the packets from the source of the data transfer as if it was the intended destination. Because the local SOS agent is deployed on the source's local network, the delay-to-acknowledgment

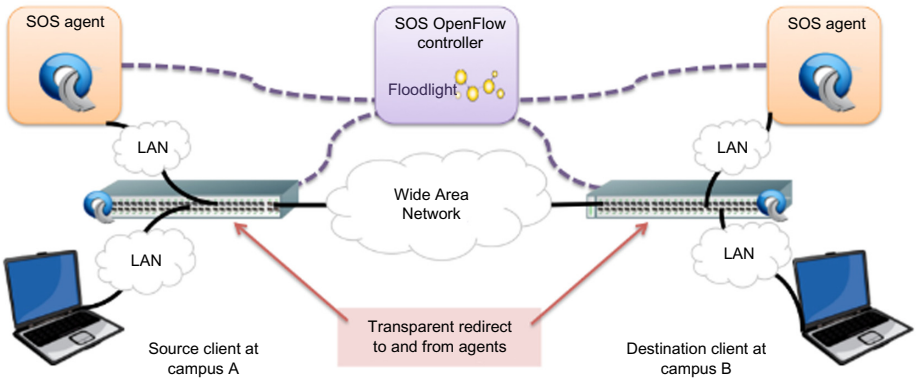

Figure 5. Steroid OpenFlow Service (SOS) network architecture. SOS works by redirecting the designated TCP connection of such a long distance data transfer to a local SOS agent via an in-path OpenFlow tunnel to a remote SOS agent.

is negligible as compared to the real destination. As such, the source can continuously transfer data to the local agent without pause.

As the source SOS agent accumulates a buffer of data from the data transfer source, a suitable destination SOS agent is located nearby the intended data transfer destination. The source and destination SOS agents establish a number of parallel TCP connections between each other to rapidly transfer the data from the source SOS agent to the destination SOS agent. The single buffer of data at the source SOS agent is fed into the multiple parallel TCP connections. The parallel TCP connections each have the same TCP windowing problem; however, since they all operate independently, one connection's pause for an acknowledgment does not prevent any of the other connections from transmitting data. In this way, the large network between the two agents can be utilized to its full potential, and the bottleneck then migrates toward the network bandwidth itself rather than TCP.

Data Storage

Advanced data transfer networks that quickly control the flow of genomics data are critical pieces of the Big Data analysis puzzle. Of course, large storage footprints are required and often guide the scale of an experiment. Using today's commercial technology, the interface of the network at the storage system is a core bottleneck in the data transfer process. This section discusses file system considerations in the context of data transfers.

As of this writing, there are two devices typically used for active data storage: magnetic disk (HDD) and solid state disk (SSD). HDD still costs considerably less than SSD and is generally used for large-scale storage. HDD have various performance rates depending on the technology. Some of the better enterprise disks maintain a read throughput of about 225 MBytes/s and a write throughput of about 222 MByte/s. These values assume that data is read and written in large sequential chunks. Typical SSD performance is about 498 MB/s reading and 458 MB/s writing written sequentially. Accessed in random 4K chunks, this drops to 31 MB/s reading and 84 MB/s writing; however, if about 64 different files are written in 4K chunks, this goes up to 204 MB/s reading and 217 MB/s writing. Hardware vendors are continuing to ramp up SSD performance so that these numbers may soon shift upward.

While a faster data transfer option, SSDs have a much shorter expected lifespan than HDD, ie, as little as 10 months in continuous usage. For this reason, most data center class storage systems still employ HDD as the primary active storage. SSD has a number of uses, though. First, SSD arrays are being pushed into production as Data Transfer Nodes (DTN) in a data center. These fast, parallel storage systems are placed close to the fastest world-facing pipe and used as a staging area for Big Data receipt prior to pushing onto slower storage. Second, medium-size databases with a write-some/read-many usage patterns work well on SSD. Another approach is to use SSD as a burst buffer – a place to write data temporarily before either being sent over the network or before being written to disk. This setup is similar to a disk cache, but regular cache is volatile whereas data on SSD survives a crash or loss of power. These burst buffers are typically managed by system software and may not be directly visible to the user. Alternatively, these can be set up for specialized purposes or specific algorithms.

In a cluster-based processing system, especially one with many users and various applications, storage configuration can be a very complicated problem. Different applications may present different workloads to storage, and multiple users may run various applications at the same time. A configuration that works really well for one workload may not work well for another. Even worse, an application that is a good match to the storage configuration may run poorly when other users execute applications that are not well suited for that workload.

Regardless of target workloads, a storage system for a large multiuser cluster will employ many storage devices (HDD or SSD, from here on we will simply refer to primary storage as "disk"). Multiple disks are employed both to achieve a large storage area and/or to improve performance through parallelism. Parallelism is simply the idea that multiple disks (and controllers, channels, server, etc.) can be operating simultaneously and thus get more I/O done per unit time. This implementation is used in almost any large-scale cluster. Not as obvious, though, is how this parallelism is used by a given workload, and to what extent the file storage system software supports this usage pattern.

Big Data transfers require parallelism. For example, a traditional logical volume manager (LVM) treats a group of disks as if they were one disk, stacked one after the other. Thus if you write large amounts of data to it from the network, it will fill up the first disk and then move on to the second, and so forth. This implementation is a great solution for managing the size of storage system, but not so great for parallelism. The likelihood is that only one disk will be accessed at a time by any given program. Note that this description does not refer to any particular LVM, as most modern products include many ways to manage the disks including redundant array of disks (RAID) and just a bunch of disks (JBOD). A different approach is striping, whereby a group of disks are also treated as a single disk, but the logical order of the data is rearranged so that one block of data (usually 4K) is written to a disk, then it moves to the next disk, and so on, in round-robin fashion. With this organization it is now much more likely that an application will access data on multiple disks at one time (as long as it accesses more than one block at a time) and thus can achieve higher performance through parallelism.

These schemes can be very useful for small to medium sized systems. They are fairly easy to manage and inexpensive to implement. On a large system there is a bottleneck. One must consider how the various nodes of the cluster will access these disk systems. Storage systems are connected to the nodes used

for computation through a network. This connection may be the main LAN network of the cluster (such as a large Ethernet, Infiniband, or Myrinet network), or this may be a specialized storage area network (SAN) designed for connecting storage (eg, fiber channel). SANs can be very expensive to implement on a large cluster, and can even be rather expensive on smaller systems due to the specialization of the hardware. Still SANs may be used within the storage system to share drives. From this point on, we will focus on LAN approaches. The main feature of a LAN approach is the use of a computer known as the "file server". This computer has the disks attached directly or through SAN and have a connection to the LAN. The file server utilizes software to allow other computers to mount its disks for access. A file server may be as complex as a large computer, with lots of memory, I/O devices, network interfaces, and several cores. By contrast, a file server may be a very simple device that runs an embedded server software such as network-attached storage (NAS), which is an appliance with disks that connects to a LAN and uses network file system (NFS) or similar to serve workstations. A large server can manage more disks, but there are limits to having all I/O requests come through one of these. While large servers can move an impressive amount of data from disk to network, the latency of each data request can overwhelm the server when a large number of small requests occur. Thus most data centers employ many servers in order to once again achieve parallelism between them.

With multiple servers, once again, how the storage system is configured can have a massive effect on performance. Each server can present a distinct file system, and parallelism occurs when users put files on different ones. This implementation is generally confusing to users, so the favored alternative is the single name space model, where it appears that there is one large file system that is actually spread across different servers. Individual datasets (files) can be placed on one server at a time, or distributed across many or even all of the file servers. Small files tend to work better being stored at one place, whereas very large datasets can be accessed in parallel if they are distributed among many file servers. Once again, we have the same problem discussed above in that some applications may not have been designed for a given storage architecture. The more servers you have, however, the more flexibility to manage how and where data is stored for the best performance. On the other hand, more servers means more cost – especially if each server is a large system with lots of RAID hardware and disks. Smaller and simpler servers managing fewer disks favor a larger number, but each individual is less powerful. In a nutshell, when designing a system for rapid Big Data transfer onto disks, one must carefully consider these aspects of storage system architecture. If not, a fast input network may be bogged down by a traditional storage configuration.

Data Staging, Archiving, and Access

In a large system similar to that shown in Figure 1 for genomics data processing, there are several places where storage systems play a key role. At the center of such systems is a repository – a place to store datasets that have been or will be processed in response to requests across the network. To the upper left is data ingest where new datasets produced by the gene sequencer hardware is brought into the system and stored in the repository. Along with the raw data is *metadata,* which includes what the data is, where it was generated, and other factors. This data may be stored with the raw data in the same dataset, near the data in its own file, in a database system, or possibly distributed in several places. Ingested data typically goes through some form of ingest processing – a standard set of codes that are applied to all data in order to get it ready for users.

Next, requests arrive for the data to be processed by some application or experiment. The data must be retrieved from the repository with its metadata and sent to a processing system, which is typically an HPC cluster of some kind. The data may need to be transmitted over a WAN first, but in either case it ends up on a temporary *scratch* volume, which is often a shared finite resource. Scratch volumes are normally optimized to provide the best performance for the applications on the processing cluster and not for long-term storage of the data. Typically, the processing system and its scratch volume utilize scheduling to control access to processors, storage, and the LAN. Processed data is stored back to the scratch volume. Once data is processed, it moves to an analysis phase, which may be carried out on the same cluster, or it may be moved to one that is better configured for this phase. Various algorithms are employed, and visualization is often a part of this phase. Data may be accessible from the scratch volume, or it may require moving. Finally, data product results are moved to a storage facility for keeping. This task may mean moving the results back to the original repository, or to a similar storage area near the processing system. Clearly, the actual data movement rate into and out of repositories is an important consideration for Big Data and is subject to the same data transfer laws described above.

A significant question one must answer is: how long must huge genomics datasets be retained?

If a central organizational repository exists, perhaps all data can remain online until it passes to the destination repository, and then removed from the system. If the scientist needs the data in the future, it can be retrieved (eg, submission to a campus archiving system, data submissions to NCBI). If data must be retained longer, how long is enough? There is a cost involved in "retaining all raw data", and this will drive the repository phase of the genomics data lifecycle. The scientist must compare this cost with the cost of simply recreating the data. Data that is being retained but with a low likelihood of access can be archived on local tape systems, cloud-based solutions such as Amazon Web Services (AWS) Glacier,[35] or Google nearline storage[36] (still in beta).

If the scientist stores the data on tape, a data management system must be put in place in order to track, monitor,

and maintain the data. Open source and commercial systems exist that can provide a file system or object store view of data with the storage of data existing on disk, tape, or cloud (or a combination). Tape, though considered "dead" at one point, continues on. Tape has high density (up to 8 PB/rack), cheap media costs, and low power use. Time to first byte is around 90 seconds, so with large genomic files this is a small percentage of the total data transfer time. Combining tape with a disk cache can ameliorate this issue, and the researcher can trade off cost for the disk cache for lower latency to access files on tape. The reality of data erosion on specific media over time is real, and data must migrate every few years to new media (whether tape or disk) to maintain data integrity.

Cloud solutions are attractive repository solutions, but the scientist must be careful to consider acceptable transfer times from sequencing instrument to the storage cloud as discussed above. Cloud service costs are transparent (eg, AWS is currently at $0.01/GByte/month or $120/TByte/year); however, there can be costs to retrieve data, so these archives could be function as write once, read rarely, and perhaps are not touched again until they are disposed after a few years. Additionally, cloud storage implementations may have annual hosting fees. Large organizations managing petabytes of data will likely choose a commercial cloud as a full or partial solution, but this decision is accompanied by the transfer of data security to the cloud provider and realities of using an external service provider including the possibility that the provider could go out of business. Just as moving petabytes of data from organization to organization across networks requires careful thought and engineering, so too does managing "static" archives of petabytes of precious biological measurements, with strong potential for exabytes of storage in coming years. The total cost of the cloud may well be lower than organizational hosting charges, but value exists in evaluating these costs for the scientist's own particular situation with geometric dataset growth in mind.

Conclusions

While moving "small data" is easy across the commercial Internet, moving and managing Big Data at the speeds needed for high-throughput science requires careful planning and engineering. This planning can result in orders of magnitude difference in transfer efficiency. Significant delay in Big Data transfer can make the difference between choosing whether to proceed down a novel research path or letting the hardware decide the amount of research that can be done in a particular timeframe. An experiment that might require a month to transfer the data might be declined, whereas a 3-day data transfer could be completed over a long weekend and research completed the following week. Technologies such as SDN, the Science DMZ, high-performance networking, and high-performance storage all play a crucial role in accomplishing this mission.

Acknowledgments

We would like to thank the National Science Foundation for funding of our research efforts through these awards: #58501934 (JRB), #1245936 (KCW, FAF), #1447771 (WBL, FAF), and #1443040 (FAF, KCW). This research was supported in part by the Intramural Research Program of the NIH National Library of Medicine (DP).

Author Contributions

Conceived and designed the experiments: FAF, JB, JD, RI, CAK, WL, DP, KW. Analyzed the data: FAF, JB, JD, RI, CAK, WL, DP, KW. Wrote the first draft of the manuscript: FAF, JB, JD, RI, CAK, WL, DP, KW. Contributed to the writing of the manuscript: FAF, JB, JD, RI, CAK, WL, DP, KW. Agree with manuscript results and conclusions: FAF, JB, JD, RI, CAK, WL, DP, KW. Jointly developed the structure and arguments for the paper: FAF, JB, JD, RI, CAK, WL, DP, KW. Made critical revisions and approved final version: FAF, JB, JD, RI, CAK, WL, DP, KW. All authors reviewed and approved of the final manuscript.

REFERENCES

1. CERN. 2015. Available at: http://home.web.cern.ch/about/computing.
2. CMS_Collaboration. A new boson with a mass of 125 GeV observed with the CMS experiment at the large hadron collider. *Science*. 2012;338(6114):1569–75.
3. Denoeud F, Carretero-Paulet L, Dereeper A, et al. The coffee genome provides insight into the convergent evolution of caffeine biosynthesis. *Science*. 2014; 345(6201):1181–4.
4. Dabydeen SA, Kang K, Díaz-Cruz ES, et al. Comparison of tamoxifen and letrozole response in mammary preneoplasia of ER and aromatase overexpressing mice defines an immune-associated gene signature linked to tamoxifen resistance. *Carcinogenesis*. 2015;36(1):122–32.
5. Sanger F, Coulson AR. A rapid method for determining sequences in DNA by primed synthesis with DNA polymerase. *J Mol Biol*. 1975;94(3):441–8.
6. Kodama Y, Shumway M, Leinonen R. The sequence read archive: explosive growth of sequencing data. *Nucleic Acids Res*. 2012;40(Database issue):D54–6.
7. SRA. *NCBI Short Read Archive*. 2014. Available at: http://www.ncbi.nlm.nih.gov/sra.
8. Wilks C, Cline MS, Weiler E, et al. The Cancer Genomics Hub (CGHub): overcoming cancer through the power of torrential data. *Database (Oxford)*. 2014;2014. p. bau093.
9. TCGA. 2015. Available at: https://tcga-data.nci.nih.gov/tcga/.
10. BaseSpace. *Illumina Basespace*. 2014. Available at: https://basespace.illumina.com.
11. *Internet2*. 2015. Available at: http://www.internet2.edu/.
12. GENI. 2015. Available at: https://www.geni.net/.
13. Foster I. Globus online: accelerating and democratizing science through cloud-based services. *IEEE Internet Comput*. 2011;15(03):70–3.
14. Aspera. *Aspera*. 2014. Available at: http://asperasoft.com/.
15. LeDuc R, Vaughn M, Fonner JM, et al. Leveraging the national cyberinfrastructure for biomedical research. *J Am Med Inform Assoc*. 2014;21(2):195–9.
16. XSEDE. 2015. Available at: https://www.xsede.org/.
17. Goff SA, Vaughn M, McKay S, et al. The iPlant collaborative: cyberinfrastructure for plant biology. *Front Plant Sci*. 2011;2:34.
18. OSG. 2015. Available at: http://www.opensciencegrid.org/.
19. Crosswell LC, Thornton JM. ELIXIR: a distributed infrastructure for European biological data. *Trends Biotechnol*. 2012;30(5):241–2.
20. Lampa S, Dahlo M, Olason PI, Hagberg J, Spjuth O. Lessons learned from implementing a national infrastructure in Sweden for storage and analysis of next-generation sequencing data. *Gigascience*. 2013;2(1):9.
21. Afgan E, Chapman B, Jadan M, Franke V, Taylor J. Using cloud computing infrastructure with CloudBioLinux, CloudMan, and Galaxy. *Curr Protoc Bioinformatics*. 2012;Chapter 11:Unit1119.
22. BGI-CLOUD. 2015. Available at: http://bgiamericas.com/data-analysis/bgi-cloud/.
23. CloudLab. 2015. Available at: https://www.cloudlab.us/.
24. ESNET. *ESNET*. 2015. Available at: http://fasterdata.es.net/home/requirements-and-expectations.
25. PERFSONAR. *PerfSONAR*. 2015. Available at: http://www.perfsonar.net/.

26. Dart E, Rotman L, Tierney B, Hester M, Zurawski J. The science DMZ: a network design pattern for data-intensive science. *Sci Program*. 2014;22(2):173–85.
27. Allcock W, Bresnahan J, Kettimuthu R, et al. The globus striped GridFTP framework and server. Paper presented at: Proceedings of the 2005 ACM/IEEE SC05 Conference (SC05); 2005; Seattle, WA.
28. Monga I, Pouyoul E, Gu C. Software-defined networking for big-data science – architectural models from campus to the WAN. Paper presented at: Proceeding of IEEE/ACM Supercomputing Conference; 2012; Salt Lake City, UT.
29. ARSTECHNICA. 2015. Available at: http://arstechnica.com/security/2012/06/26/science-dmz/.
30. Benson T, Akella A, Maltz D. Unraveling the complexity of network management. Paper presented at: Proceedings of 6th USENIX Symposium on Networked Systetms Design and Implementation, ser. NSDI'092009; Boston, MA, USA. 2009.
31. McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comput Commun Rev*. 2008;38(2):69–74.
32. Wang G, Ng T, Shaikh A. Programming your network at run-time for big data applications. Paper presented at: HotSDN '122013; Helsinki, Finland. 2012.
33. Rosen A, Wang KC. Steroid OpenFlow service: seamless network service delivery in software defined networks. Paper presented at: First Annual GENI Research and Educational Experiment Workshop2012; Los Angeles, CA, USA. 2012.
34. Das A, Lumezanu C, Zhang Y, Singh V, Jiang G, Yu C. Transparent and flexible network management for big data processing in the cloud. Paper presented at: 5th USENIX Conference on Hot Topics in Cloud Computing, ser. HotCloud'13.2013; Berkeley, CA, USA.
35. Amazon. *Amazon Glacier*. 2015. Available at: http://aws.amazon.com/glacier/.
36. GoogleCloudNearline. *Google Cloud Storage Nearline*. 2015. Available at: https://cloud.google.com/storage/docs/nearline.