Give Your Data the Edge: A Scalable Data Delivery Platform

1 Introduction

Scientific collaboration is increasingly data driven. Large volumes of data are generated, aggregated, archived, and shared—often with collaborators (and their compute resources) spread across the globe—with subsequent analysis generating still more data to archive and distribute [**1**]. Scientists either become experts at data transfer, storage, and management, or their ability to build on each other's work suffers. This project addresses this challenge, with an emphasis on easing the burden on the user, building a solution that is sustainable over the long term, and delivering performance that scales in the number of collaborators.

We are deploying a general-purpose storage platform, called *Syndicate*, that harnesses a collection of available storage components to provide a global, scalable, and secure storage service. These include public and private cloud storage (for data durability), network caches and content distribution networks (for scalable read bandwidth), and local disks (for local reads/write performance). The goal is to make it possible for applications to *access data independent of where it is stored,* where the storage platform: (1) minimizes the operational burden imposed on users, (2) maximizes the use of commodity infrastructure; and (3) maximizes aggregate I/O performance.

Existing systems typically focus on performance bottlenecks, but often do so in a way that puts management and operational burdens on the user [**2**,**3**]. In contrast, our approach addresses performance issues as unobtrusively as possible, by automatically combining existing components into a coherent, multi-tier storage hierarchy that is easily integrated into the user's current application workflow:

- **Cloud-Ready –** Allow users to mount shared volumes into cloud-hosted virtual machines (VMs) with minimal operational overhead. To support this, Syndicate leverages a multi-tier storage hierarchy to provide a traditional read/write file system rather than offering a user-managed staging solution.
- **Scalable Read Bandwidth –** Provide scalable read bandwidth (i.e., supports a scalable number of users) with minimal operational overhead. To support this, Syndicate transparently uses commodity CDNs—the cornerstone of scalable Internet video delivery—rather than require users to manually stage data or manage a set of mirrors.
- **Provider Independence –** Allow users to take advantage of cost/performance tradeoffs among multiple storage providers (as well as spread risk across those providers) with minimal operational overhead. To do this, Syndicate treats both commodity and private cloud storage as a virtual block storage device.
- **Secure-by-Default –** Allow users to securely share files across organizational boundaries, at scale, with minimal operational overhead. To do this, Syndicate provides a declarative security policy that fully automates key distribution and certificate management.
- **Adapt to Existing Workflows –** Make it easy to integrate existing user workflows, datasets, and toolkits, as well as extend and customize to meet specific community requirements (e.g., privacy). To do this, Syndicate is structured to accommodate domain-specific plug-ins (called *drivers*).
- **Sustainable Design –** Provide a general-purpose storage platform that leverages commodity storage and network caches at every opportunity. To do this, Syndicate offers a logical storage service that takes advantage of building blocks already deployed throughout the Internet.

We start with a target set of scientific user communities and the challenges they face. These are introduced in Section 2. Syndicate addresses these challenges by combining existing authentication services, cloud storage, content distribution networks (CDNs), and public datasets to create a coherent

read/write storage layer on top of existing infrastructure. It also adopts a modular design that makes it easy to customize the system for different usage scenarios. Section 3 sketches Syndicate's architecture.

We have implemented and evaluated Syndicate on a collection of scientific workloads, as reported in Section 4. We conclude Syndicate provides value, in large part because the scientific computing often includes datasets and workflows with domain-specific storage requirements, which make it difficult to leverage commodity cloud infrastructure. These include operational requirements like preserving workflow compatibility across different storage systems, policy requirements like mandatory authorization and end-to-end data confidentiality, and the pragmatic issue of archived datasets being remote from available computational resources. As such, many labs continue to maintain their own storage infrastructure to meet these requirements, even if the cost of doing so is higher than using available 3rd-party resources. Syndicate changes the model by making it easy to use commodity resources in a domain-specific way.

Syndicate's value proposition is to fully decouple storage semantics from infrastructure. This lets users select infrastructure based on its cost/performance trade-off, while ensuring that their domain-specific storage requirements are met. Our plan to build and operate a pilot deployment of Syndicate spanning edge resources on nine pilot campuses, managed by a unique team with a long track record delivering advanced cyberinfrastructure to the research community.

2 User Communities

Our approach can be adapted to a wide range of scientific datasets and workflows, but our plan is to focus on a specific pilot that demonstrates the feasibility and value of our design. This section identifies the scientific communities that will participate in the pilot, and summarizes the storage challenges they face.

2.1 iPlant Collaborative

The life sciences have benefited greatly from massively parallel and highly quantitative technologies, resulting in researchers amassing datasets that are a challenge to analyze and interpret in a timely manner. The iPlant Collaborative is an NSF funded comprehensive national cyberinfrastructure (CI) resource that addresses this challenge [**4**].

The iPlant CI provides a web accessible tool-set for performing data analysis and managing data-driven collaborations, with an emphasis on federating data and consuming computational resources from multiple providers. These include NSF-funded XSEDE resources, private campus clusters, and commercial cloud providers. The iPlant CI provides an avenue for researchers to share their research data, software tools, and analysis pipelines with their collaborators and/or a large community of users without burdening the researchers to provision this diverse underlying computational infrastructure. In addition, iPlant provides life scientists with the ability to securely manage their data and perform complex analyses using software tools that are configured and optimized for various high performance platforms.

At the center of iPlant CI is its iPlant Data Store (iDS), built on top of iRODS [**5**]. It provides the foundation for scalable data management that multiple communities use every day. Here, we focus on four specific use cases that heavily leverage iPlant CI to meet their science needs.

- Professor Edgar Spalding at the University of Wisconsin has developed novel robotic imaging systems for high throughput imaging of plants. This infrastructure generates large volumes of data for community members that send samples to this facility. Multiple groups access these data sets to perform analysis using distributed computing methods [**6**]. The journey of data starts in Wisconsin, uses iDS storage primarily in Arizona (taking advantage of iDS data management capabilities), and is analyzed at many different sites (taking advantage of Condor pools available through OSG and Discovery Center at Wisconsin). This distributed access and high throughput demands for edge computing have resulted in multiple bottlenecks, with demand for this style of collaboration continuously growing as campuses across the US collaborate through the Genome-to-fields. *The*

challenge is to transparently and efficiently make widely distributed datasets accessible from the widely distributed computational resources that researchers have available to them.

- Professors Nilima Sinha and Julin Maloof at UC Davis share over 250TB of data with their collaborators through iDS, making use of Atmosphere cloud platform and XSEDE resources at TACC for managing their distributed analysis pipelines. The data contains images, next generation sequencing (NGS), and derived data products from their large-scale analysis. Mounting this data into application processing using FUSE [**7**] is a common model for this research community, as it makes it easy to use existing applications and 3rd party software packages. *The challenge is to make data readily available for widely distributed computational resources without changing the application workflow and toolset.*
- Professor Paul Flikkema at Northern Arizona University has established an NSF-funded Southwest Experimental Garden Array (SEGA), a new genetics-based climate change research platform that allows scientists to quantify the ecological and evolutionary responses of species to changing climate conditions. SEGA comprises 10 gardens along the elevation gradient in northern Arizona, and because temperature and moisture predictably change with elevation, these gardens reflect climatic differences from desert-to-alpine-forests that mimic the effects of climate change. SEGA has multiple sensors that stream data from remote locations over unreliable network to NAU, and then onto iDS. *The challenge is to make data generated at remote sites readily and reliably deposited into data repositories without having to alter the acquisition pipeline.*
- Professor Gwen Jacobs at the University of Hawaii is collaborating with iPlant to establish a remote node for iPlant to allow researchers at her institute access to iDS. The campus relies on many commercial and academic cloud services for data storage and analysis. This complements that campus resources at Hawaii, which is important due to costs the campus having limited footprint for computing infrastructure. Network throughput and latency are common bottlenecks that throttle research. *The challenge is to make remote data sets and transfer of data and analysis more responsive when using commodity and remote computational resources.*

2.2 M-Lab Consortium – Internet Traffic Analysis

Professor Peterson is a founding member of Measurement-Lab (M-Lab), a collaborative effort to capture, archive, and analyze Internet traffic data. It includes a distributed server platform on which researchers can deploy open source Internet measurement tools, with the data collected by those tools released in the public domain [**8**]. The goal of M-Lab is to advance network research and empower the public with useful information about their broadband and mobile connections. By enhancing Internet transparency, M-Lab's goal is to help sustain a healthy, innovative Internet.

A joint project of the PlanetLab Consortium, Google, the New America Foundation, and a proactive set of academic researchers, M-Lab is now a global platform with over 130 servers across every continent except Antarctica, involves over 35 corporate and governmental partners, supports 16 different performance tools, runs over 200k tests every day, has led to over 25 published research papers, and it has been used to collect over a petabyte of measurement data. Among the many stakeholders that benefit from M-Lab, governmental regulators and policymakers gain access to objective broadband data. For example, the US Federal Communications Commission (FCC) has partnered with M-Lab to study broadband coverage and performance data across the major US service providers. Analogous national commissions are conducting similar studies in other countries.

The data collected by the M-Lab platform and tools is archived in a data repository hosted in Google Cloud Storage. The repository currently has 1,498,005 files and 13,055 directories, and is growing linearly as a function of time [**9**]. The current workflow involves manually staging the data to a scalable number of virtual machines (VMs) running in a public or private cloud. *The challenge is to make M-Lab data automatically available to cloud VMs as a mountable volume.*

2.3 Clinical Trials for Personalized Medicine

Professors Hartman (CS) and Hurwitz (Med School) at the University of Arizona are building a cloud service to support a clinical study of diabetic foot ulcers, with the goal of personalizing treatment. The big data aspect of the trial is interesting because it involves a meta-genomic database that spans patient-to-population genomes. Meta-genomics is the comparison of genetic material in different samples, without isolating the DNA from different species or even different individuals [**10**]. The project has the potential to have a significant impact on an important healthcare issue. Chronic wounds represent a significant clinical burden in the US and directly impact the quality of life for patients. It is estimated that the treatment of DFUs and associated amputations alone costs over 17 billion dollars per year. Moreover, more than 60,000 limb amputations are performed each year as a result of chronic DFUs that are unresponsive to treatment, leading to high costs for hospitalization and increased morbidity and mortality. Delay in diagnosis can also lead to rapid progression in infection and result in amputation. Currently, diagnosis is only possible using standard clinical methods in 20-50% of DFUs. Developing strategies for faster diagnosis, directed treatment of microbial infection, and rapid wound healing is critical for effective patient care and decreased costs associated with treatment.

The meta-genomics analysis involves performing pair-wise comparisons between samples. The size of each is on the order of 100MB, and we anticipate thousands of samples in the study. The bulk of the analysis pipeline is being implemented using Hadoop and HDFS [**11**], and there is a need to bring the data sets from the iRODs repository on which they are stored to the Hadoop cluster on which the computation is performed. Currently, the analysis requires manually staging the desired samples to a private cluster, performing the analysis using a custom application, and de-staging the results back to iRODs. *The challenge is to integrate HFS with remote data sources so that data movement happens automatically—necessary data sets are pulled to the proper Hadoop nodes, the resulting data are pushed back to iRODs, and datasets are automatically cached for use in subsequent computations.*

2.4 Retail and Consumer Analysis

Professor Anita Bhappu and collaborators at Arizona conduct research on customer interactions and service delivery by doing data mining on customer survey data and customer transaction data provided by corporate partners. Ensuring privacy is paramount in this work due to the high risk of individual subjects being personally identified. This risk, in turn, stifles the ability to collaborate. Being able to share data across a "secure data commons" with academic and industrial partners would promote better understanding of consumer behavior. *The challenge is providing a secure data sharing substrate that is also efficient and does not burden researchers with onerous security mechanisms.*

3 Syndicate Architecture

Syndicate is a platform for building a scalable storage service out of existing public and private storage components. A robust prototype exists and has been evaluated on several of the workflows and datasets identified in the previous section. This section describes Syndicate's architecture (Figure 1), focusing on how it lowers the barrier-to-adoption by different user communities and delivers scalable performance.

To address the first goal—minimize operational overhead—Syndicate's storage layer is designed to be self-managing and as easy to deploy as the application it serves. Syndicate does this by implementing the storage layer as a set of processes, called *Syndicate Gateways (SG),* that are co-located with application processes (e.g., they are included in the application VM image). These gateways discover and coordinate with one another via a scalable Syndicate *Metadata Service (MS)*.

To address the second goal—scalable performance—the distributed gateways leverage commodity CDN and cloud infrastructure, but do so in a way that preserves application-defined storage semantics without coupling the application to specific providers. Gateways work together with the MS to make data available to applications in a *Volume* that can be securely mounted into all of the application's processes.

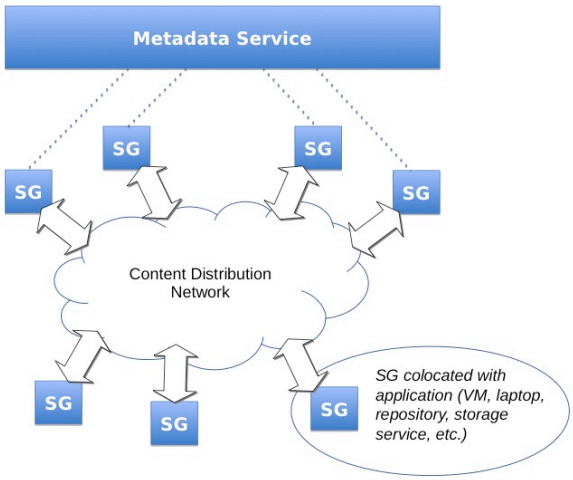

Figure 1. Syndicate Gateways (SGs) are co-located with applications and leverage CDNs to scalably transfer data, all under the control of a scalable Metadata Service (MS) running in Google App Engine.

The MS is a scalable "always-on" service that runs in a Google App Engine [**12**], a commodity Platform-as-a-Service (PaaS). It hosts the authoritative copies of all volume and file metadata, the deployment's configuration, and the deployment's current set of users. We place these responsibilities in a centralized service apart from gateways to ensure that the system can make forward progress regardless of the gateways' transient availability. Figure 1 gives a high-level depiction of Syndicate.

3.1 Correctness

The MS and SGs collectively implement a set of protocols that automatically mask failures, distribute load, securely deploy new driver code and configurations, and enforce gateway membership in a volume and Syndicate instance. These protocols allow the system to work even if each gateway is behind a NAT, making it feasible to deploy application code on any machine from cloud VMs to personal computers.

Syndicate's consistency model is similar to that of today's intra-cluster distributed filesystems, making it straightforward to integrate with existing workflows expecting such storage systems [**13**,**14**,**15**,**16**,**17**]. Its consistency protocol divides a volume's files into blocks, and provides per-block delta-consistency [**18**] on reads (with a per-file delta which, when set to zero, yields sequential read consistency), even when using commodity CDNs that assume cached data is immutable. Overcoming the CDN's weak consistency is critical to correct operation [**19**,**20**]. Syndicate does this by accessing individual blocks via HTTP URLs, and on processing a write, the gateway changes the addressing information for a block (i.e., a new manifest for the modified file) and propagates it to the MS. Other gateways fetch and cache this information from the MS at application-specified times, and use it to generate URLs that refer to fresh versions of the file's blocks in the CDN. The CDN simply treats different versions of the same block as different objects since they identify Web objects by URL.

Writes to a file in Syndicate are sequentially consistent by default. Each file has a designated "coordinator" gateway—initially the gateway that created it—that aggregates write metadata messages from other gateways, serializes them, and sends them off to the MS. Gateways independently replicate blocks to back-end cloud storage, but the coordinator decides which blocks constitute the current state of the file. This allows Syndicate to scale write-bandwidth in the data plane since it performs write ordering separately from write replication.

In addition to using our consistency protocol for distributing fresh data, gateways use it to distribute global configuration information at scale via the CDN. Gateways learn when the configuration has changed by sending their latest-known configuration's version in-band with data-plane messages. An SG will be obliged to reload its configuration when it notices this version has increased, since the other SGs or the MS will NACK any of its subsequent requests that do not include the latest value. This ensures

that gateways always learn when the configuration changes before they can process reads and write. Once a gateway detects a new version, it uses it to generate URLs for the latest copies of the configuration data and synchronously pulls them from the MS through the CDN.

3.2 Scalable Performance

An evaluation of Syndicate's consistency protocol shows that it works as desired. The availability of edge caches allows a scalable number of readers to always pull fresh data from other sites, even when modifying shared datasets. However, once our prototype removed the data-plane bottleneck, the rate at which the MS could process metadata requests became the new bottleneck. This is in spite of the fact that unlike the data-plane, Syndicate's file metadata entries have a small, constant size.

The fundamental problem was that the aggregate storage bandwidth in the MS needed to scale along with the number of readers and writers. We noticed that our algorithm for listing a directory did not perform as expected under the targeted workloads, specifically, for and M-Lab, *where a directory can have thousands of children.*

To overcome this problem, we created a new directory-indexing algorithm that scales with the number of file metadata records and concurrent I/O requests, all the while leveraging a commodity PaaS. Combined with using cloud storage and CDNs for the data plane. Our efforts mean that the user does not need to set up and manage any dedicated storage infrastructure to create a scalable domain-specific storage system. He or she just leverages services already running in the commodity cloud.

3.3 Secure Automount

Syndicate's security model ensures that a malicious adversary cannot undetectably tamper with messages, blocks, and configuration state while they are being transferred, cached, or stored. At the same time, Syndicate grants each gateway a set of volume-wide capabilities—whether or not it can read, write, and coordinate for files.

To enforce these properties, Syndicate cryptographically binds each gateway to exactly one volume, exactly one principal, and exactly one set of capabilities. Each gateway is assigned its own public/private key pair, and is configured to trust a volume-specific public key whose private counterpart resides on the MS. The public key and capabilities for each gateway are signed by the volume's private key and distributed to a scalable number of gateways via the CDN (i.e. as part of the global configuration state), thus ensuring that once a gateway processes the latest configuration, it will know which public keys to trust. Gateways sign messages they send to each other and the MS, as well as each block they distribute and replicate, allowing them to block unauthorized operations and catch data-plane tampering.

Adding and removing users and distributing keys to gateways were a manual process in our first prototype. While this was not a huge concern for users with third-party infrastructure in place to manage tasks like these, using Syndicate in the realm of scientific computing helped us discover ways to largely automate both tasks. For example, because most scientific computing sites already have a notion of authenticating users, we added support for *authentication plugins* to the MS, thereby allowing Syndicate to authenticate users via existing user databases and existing single sign-on protocols. For example, we have successfully linked Syndicate with an external user database via OpenID [**21**].

Minimizing the operational burden of certificate distribution was more challenging. We considered using an existing certificate distribution service for distributing public keys to gateways, but found that doing so allows revocations to happen asynchronously with respect to processing writes, thereby introducing exploitable race conditions. By embedding certificate distribution in Syndicate's data plane, both certificate distribution processing and data-plane processing "share fate" with one another—if Syndicate cannot do one, it will not do the other.

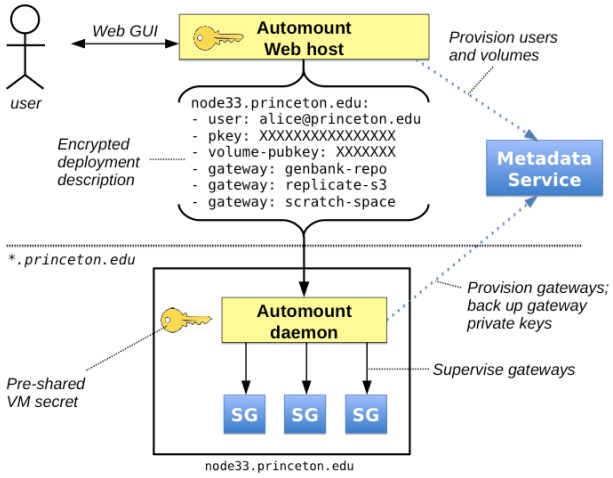

Figure 2. Overview of Syndicate's automount system. An automount daemon in each VM automatically provisions, starts, and supervises Syndicate gateways.

We solved this by creating a distributed automount service that lets VMs set up their own gateways in a secure manner (Figure 2). The automount service grants each VM a limited-privilege user account on the MS that is permitted to instantiate only the gateways specified in the deployment description. It securely instructs the VMs to generate their gateways' private keys locally (keeping them secret from all other hosts), and register the public keys with the MS. We install the automount service's public key and a pre-shared secret into the VM image before deploying, and sign its instructions to the VM's automount daemon. This allows the service to ensure its instructions' confidentiality, integrity, and authenticity, allowing users to control sets of VMs across untrusted networks.

3.4 Customization

The programming model for Syndicate is designed to minimize the effort required to address common classes of domain-specific requirements, but in a way that does not sacrifice generality. Developers specify their semantics as a *driver module* that defines arbitrary side effects to reads, writes, coordinator changes, and configuration changes. These side effects not only allow the developer to add new storage functionality, but to also add support for new back-end infrastructure. For example, the driver could have gateways replicate the write history for text files to a VCS system. By default, Syndicate offers filesystem-like semantics, and comes drivers that add back-end support for Amazon S3 [**22**], DropBox [**23**], Box.net [**24**], Amazon Glacier [**25**], Google Drive [**26**], and local storage.

Syndicate also offers developers a gateway SDK to allow them to define application-specific APIs. Developers can create their own gateway "flavors" that have the core Syndicate functionality built-in, allowing them to adapt Syndicate to existing applications and workflows. For example, our prototype ships with gateways that offer a filesystem API, a REST API, a Hadoop library, and a Python library.

Programming Syndicate is a piecemeal process by design. Domain experts "evolve" Syndicate's default semantics to fit their requirements by adding extensions to the default behavior. This facilitates experimentation and incremental development and deployment, and lowers the barrier to getting started. Note that while experts can extend-and-adapt Syndicate by writing drivers, end-users simply mount Volumes into their application processes.

4 Scientific Datasets and Workflows

Our goal is to provide a sustainable, general-purpose storage substrate—suitable for a wide-range of scientific domains—but in a way that permits each domain to customize the storage substrate to meet its specific requirements. Our experience adapting Syndicate to different scientific data storage scenarios

leads us to believe that this is achievable. This section reports this experience, and in doing so, shows how Syndicate can be used to fulfill four complementary storage roles in scientific computing.

4.1 Auto-Staging Curated Datasets

Dataset repositories are a class of scientific storage system that host curated datasets, where records tend to be written once, read many times, and rarely deleted (i.e. an append-only single-writer store). They typically host data from previous experiments or data streams for future analysis. As such, they tend to host a large, ever-growing amount of data.

We have used Syndicate to minimize the operational costs of using two public repositories: the GenBank dataset from the National Institute of Health, and the M-Lab dataset curated by Google. The GenBank dataset hosts the genetic sequences from many often-studied organisms in forms amenable to analysis by the widely-used BLAST tool [**27**], and currently has a total of 271,387 files and 27,645 directories [**28**,**29**]. Its size doubles every 18 months [**30**]. The M-Lab dataset hosts a time-series of Internet bandwidth measurements gathered from vantage points across the world since January 2009, and currently has 1,498,005 files and 13,055 directories, and is growing linearly as a function of time [**9**].

The two common operational costs with using public repositories are in staging data and querying data. Because these datasets are so large, scientists cannot feasibly download them completely. Instead, they search for records within the dataset on-line, and stage downloaded copies to servers "nearby" the computers that will run the analysis. They must do so at least once for each workflow, even if only to develop automatic staging system to handle it afterwards. Moreover, each repository offers a different query API and mechanism—FTP for GenBank and the gsutil program for M-Lab—meaning that any automatic staging system must be tailored to each specific repository.

Syndicate offers a unified API for accessing these datasets, while leveraging CDNs to automatically stage records. In each case, we have a gateway in one VM import the repository records into the volume by serving as their coordinator, and having it run a repository-specific driver to interface with it. We say "import" because the driver removes the need to copy records to persistent storage, thereby keeping storage costs down. For example, a request for a block in a GenBank file would be translated into the FTP directives to open the file, seek to the block's offset, and read a block-sized chunk of data. At the same time, it asynchronously crawls the repository to translate dataset records into Syndicate files, and instructs the gateway to propagate metadata for them to the MS whenever they change to ensure reader gateways receive fresh data (Figure 3).

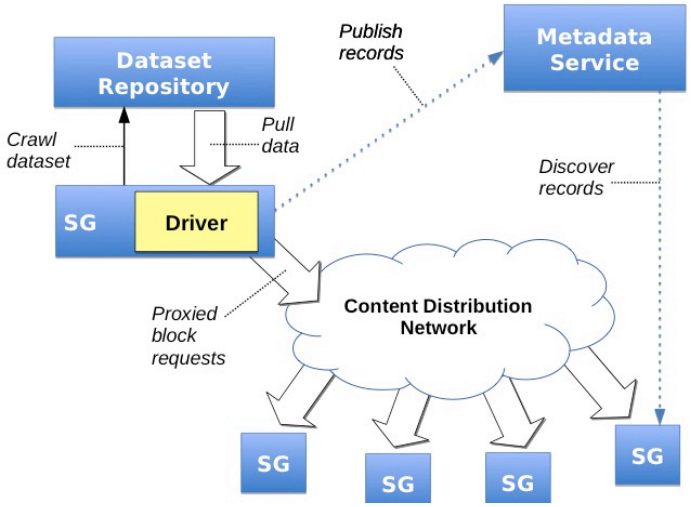

Figure 3. Syndicate co-opts CDNs as a staging area. One gateway "imports" the records of a repository by crawling it, replicating its metadata to the MS, and responding to block requests from other SGs.

The result is that Syndicate offers an automatic staging mechanism in a reusable way, and it offers a consistent, scalable way to query data in repositories. Granting the application access to the repository is simply a matter of having its designated gateway(s) mount the appropriate volume. Syndicate ensures it automatically fetches fresh data from the gateway that interfaces with the repository via the volume's designated CDNs, thereby co-opting them for temporary staging space. As a bonus, each driver can be configured to offer a consistent representation of the dataset, so that if the dataset is reorganized or relocated, the driver can be updated to preserve compatibility with local workflows.

4.2 Supporting Scalable Data Sharing

Many scientific computing projects are collaborative efforts, and offering scientists a network-shared read/write storage medium for hosting working data is a well-understood need. For example, science clouds such as the iPlant Collaborative offer users a sharable virtual block device, the Nimbus Project offers an Amazon S3 work-alike, and the Open Science Data Cloud offers shared network filesystems.

Our experience with shared storage in practice comes from using the iPlant Collaborative and private clouds (specifically, OpenStack clusters), where we encountered two key operational barriers. First, scientists need to access their shared data from outside their compute cluster of choice, but in a way that preserves data confidentiality, integrity, and authenticity. Second, scientists sometimes need to share data with many machines, for example, to deploy software to multiple clusters across the wide-area, or expose data to a grid of off-site computers.

Today, users access shared data by either remotely logging into the VM (i.e., via ssh) or by securely downloading the data (i.e., via scp or HTTPS), then use local tools to modify it, and finally upload the changes back. While these approaches are meant to ensure that only authorized users access the confidential data, they both impose operational hurdles by limiting the types of interactions. The former prevents users from leveraging tools available on different operating systems (i.e., Windows), and the latter is not only bandwidth-intensive, but also introduces the possibility that collaborators will accidentally clobber each other's changes.

Sharing data outside the site is also challenging. In iPlant, as a security feature the VMs are behind a NAT, allowing remote access only through SSH (precluding anonymous reads). It is possible to give VMs public IP address, but doing so in iPlant requires manual intervention—the VM must be moved to a separate subnet, and a port must be opened in the University of Arizona firewall. Similarly, doing so in private clouds often requires modifying the program to bind on a port in a cloud-specific manner. In general, users often must overcome various operational constraints to share data externally.

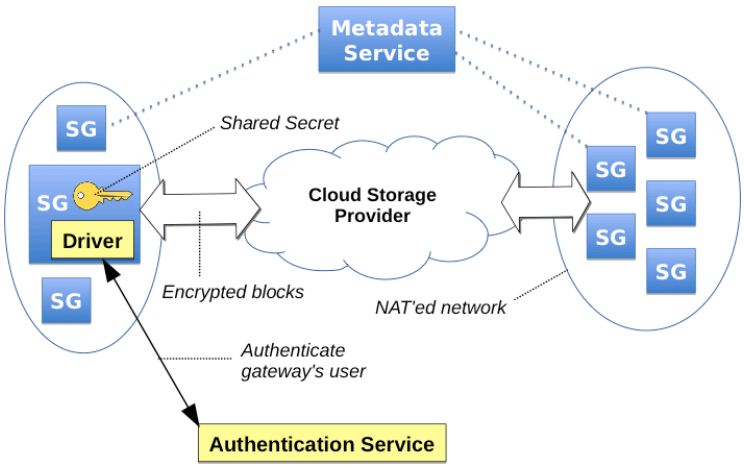

Figure 4. Syndicate helps scientists share data across using a commodity cloud storage provider of their choice. This is similar to systems like Dropbox, but also preserves domain-specific requirements.

Syndicate provides a seamless data-sharing solution that works in all three cases (Figure 4). Instead of interacting with data via network-level tools like SSH, users do so via the filesystem by creating a shared read/write volume (taking advantage of the automount service) and using our gateways to access it in each VM and personal computer. Doing this is as easy as installing Dropbox on your laptop.

If required by the application domain, Syndicate can also preserve the security guarantees offered by SSH and HTTPS. It does this using a custom driver that adds end-to-end data confidentiality by encrypting volume data with volume-wide shared secret. The driver also has the MS authenticate users with the platform via their platform-specific credentials, ensuring that only authorized users access it. We have used this driver to ensure only private cloud users can access the private cloud Syndicate deployment, for example.

Syndicate naturally overcomes the NATs in iPlant by replicating encrypted data to cloud storage. Syndicate also allows users to host data in the VM, in which case the driver ensures that the gateway in other VMs pull blocks from a CDN to reduce the bandwidth burden on the origin site. In all three cases, users interact with data with whatever hosts and with whatever tools they choose, and Syndicate provides a degree of interface and workflow consistency that were not previously available.

4.3 Augmenting Existing Data Management Systems

Instead of migrating workflows and data to cloud infrastructure wholesale, some labs prefer to augment their in-house infrastructure with third-party infrastructure as needed. We experienced this requirement with the iRODS deployments at the University of Arizona and the Texas Advanced Computing Center. Both sites already have extensive in-house infrastructure, but they are connected by an under-provisioned network link that on-site readers can congest by continuously requesting off-site data. System administrators would like to use commodity cloud and CDN infrastructure to alleviate this by caching data closer to sites and making data available through multiple routes (which may offer better bandwidth than the UA/TACC link), but cannot easily do so because it is not guaranteed to preserve iRODS' semantics.

Preserving iRODS' semantics is particularly challenging because iRODS lets users define arbitrary *rules* that control how it manages their data. This means that any attempt to augment iRODS with third-party infrastructure must preserve the semantics encoded by these user-given rules. The administrators at the University of Arizona are particularly concerned with ensuring that clouds and caches will preserve consistency-related rules.

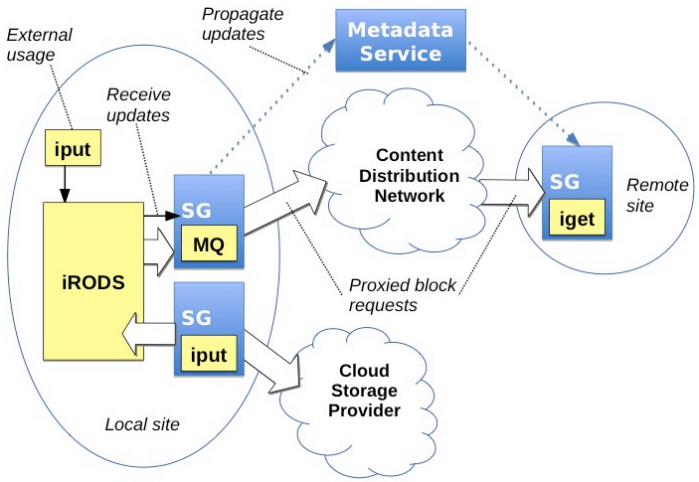

Figure 5. Syndicate ensures that remote readers see fresh data by running a gateway with an iRODS message queue driver (MQ) that listens for iRODS updates. This gateway also serves blocks through the CDN, accelerating remote readers (iget), while letting writers (iput) replicate to cloud storage.

Our solution is to encapsulate the iRODS deployment with custom Syndicate gateways that act as drop-in replacements for iRODS clients (Figure 5). Our iRODS-based drivers handle reads and writes, so that the rules will continue to be enforced. The drivers are extensible, so users can take advantage of Syndicate to replicate writes to cloud storage as well if they desire.

At the same time, we deploy on a publicly routable host a gateway that contains a driver that acts as a bidirectional proxy between iRODS and Syndicate and the cloud infrastructure it leverages. On one end, the driver receives notifications from iRODS when other clients outside of Syndicate add, modify or delete files, or alter their catalog entries. The driver replays the operations to Syndicate, thereby ensuring that the gateways retain a coherent view of iRODS data. On the other end, the gateway serves blocks from iRODS to the CDN, translating a block request into the iRODS-specific request. This lets iRODS transparently take advantage of the CDN's capacity.

The result is that the MS always has up-to-date metadata for all files accessed in the iRODS deployments, regardless of which iRODS workflow modified them. This allows us to bring to bear Syndicate's consistency protocol, allowing both sites to leverage commodity infrastructure to overcome their under-provisioned link. All the while, iRODS' rules remain in force, since iRODS is on the critical paths for un-cached reads and writes.

4.4 Enabling Cloud Orchestration

Syndicate is designed to support cloud-hosted applications (e.g., accommodate private networks and auto-mounting data into a set of VMs). But this support goes beyond enabling data access—it also supports the programs to be executed on the data. For example, we can include a copy of the latest BLAST software suite [**27**] within a shared volume holding genomics data. Then, once the volume is mounted in the VM by the automount daemon, we can extend the automount daemon to automatically run a script in the volume to kick off the workflow, using programs contained in the volume.

The value of doing this is real from a scalability standpoint. Experience shows that it can be problematic to deploy runtime state (packages, binaries, containers) to a scalable number of servers. For example, it is common for scalable compute jobs to first download a Docker container from a public repository (e.g., `docker pull afgane/galaxy101`). Once a user starts a few dozen nodes asking for the same container, the repository becomes overloaded. Having the container available in a Syndicate volume means the user can simply start the container; Syndicate already leverages network caches to automatically distribute the container without putting undue load on the repository.

5 Project Plan

The project is organized around three general tasks:

- **Deploy and Operate the Pilot –** We are deploying Syndicate across multiple sites, and operating it on behalf of a pilot user community.
- **Implement Domain-Specific Drivers –** We are integrating Syndicate into the workflow of a set of user communities by writing the necessary Syndicate drivers.
- **Enhance and Evolve the Core –** We will continue to enhance and evolve the Syndicate core based on experience supporting the datasets and workflows of the pilot user communities.

5.1 Pilot Deployment

As part of the pilot we are deploying and operating a set of hardware and software components. With respect to hardware, each participating University will install a six-node cluster in their campus Science DMZ. *Initial pilot edge (campus) sites include Arizona, UNC, Princeton, Texas, UC Davis, Wisconsin, Indiana, Hawaii and NAU.* We are augmenting these edge sites with existing research infrastructure: 70-node clusters at each of four US sites and one European site, and 2-4 node clusters at ten Internet2 routing centers.

With respect to software, the pilot includes cloud management software (the various clusters run OpenStack), a commercial grade CDN (we leverage the licensed CDN product), and the various components that make up Syndicate itself (including S3 as an example cloud storage service and Google App Engine as the hosting platform for the Syndicate Metadata Service).

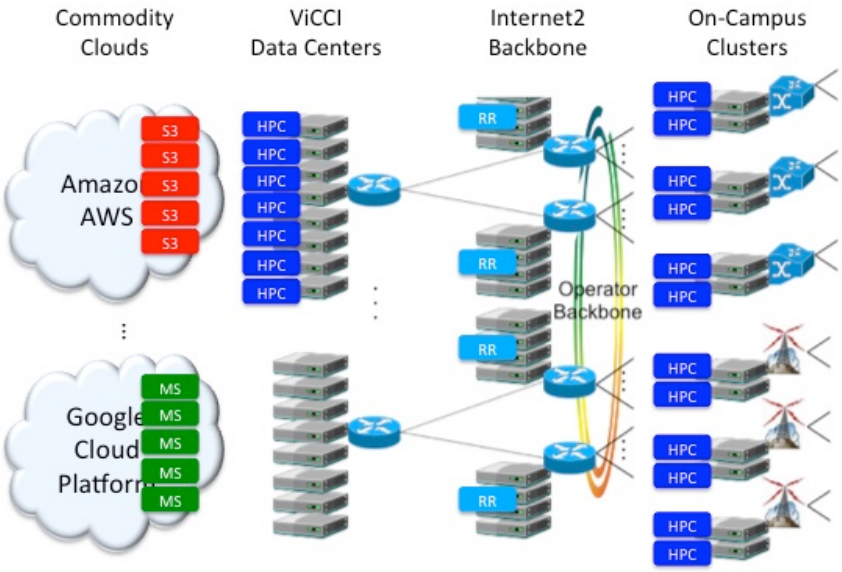

Figure 6. Pilot Deployment: Multi-tier cloud with services distributed across all four tiers.

Figure 6 depicts this combination of hardware and software as a whole. This configuration forms a multi-tier cloud that includes a combination of public/private and datacenter/edge clouds, where servers span commodity-to-datacenter-to-backbone-to-edge, with the software services deployed throughout. The following discusses four salient features of the deployment.

First, the CDN consists of a scalable caching service called HyperCache (HPC) and a scalable Request Router service (RR) that forwards requests to the best cache. Both are products that Akamai licenses to Telcos (and both correspond to technologies that the PIs originally developed on PlanetLab and subsequently commercialized). In collaboration with Internet2, we have a research agreement in place with Akamai that allows us to use these technologies in research purposes. We plan to use a two-level caching hierarchy, with the leaf caches on the campuses of participating Universities.

Second, Figure 6 shows the Metadata Service (MS), CDN (HPC and RR), and cloud storage (S3) that Syndicate builds upon, but it does not show the Syndicate Gateways (SG). This is because the SGs are co-located with the application programs, and so would run on the desktops, private clouds, and commodity clouds that host those applications. These domain-specific resources are not shown in Figure 6.

Third, all of the software used in the pilot are designed to be instantiated on Infrastructure-as-as-Service (IaaS) rather than require dedicated hardware. The pilot takes advantage of this fact, and includes a centralized service orchestration framework. This framework, in turn, leverages OpenStack to manage VMs on each of the constituent clusters. Note that starting with virtualized servers makes it easy to deploy auxiliary services. For example, we will also run perfSONAR [**31**] on each cluster.

Fourth, Figure 6 shows S3 as a representative commodity cloud storage service, but we also expect to incorporate existing private storage already in use by the pilot communities. Importing various storage platforms and data archives is one of the main activities described in the next subsection.

5.2 Domain-Specific Drivers

A key activity of this project is to extend Syndicate with the drivers needed to access datasets of value to our pilot community. We also need to connect to existing authentication services, allowing Syndicate to work with the principles and access policies already in place. In the context of the Syndicate architecture,

this translates into implementing a variety of driver modules, which we organize according to the external (to Syndicate) subsystem:

- **Acquisition Drivers –** Acquisition drivers connect Syndicate to existing data archives, allowing Syndicate to serve the content of a data source and expose its records as a directory hierarchy in a Syndicate volume. In addition, the drivers consume a source-agnostic schema from the Syndicate operator that describes how the resulting directory hierarchy is structured, and which principals are allowed to access it. We plan to implement acquisition drivers for the following data sources: (1) GenBank, (2) M-Lab, and (3) iRODS.
- **Replication Drivers –** Replication drivers connect Syndicate to existing durable storage, providing Syndicate with an abstract "block device." However, there are cases where a more expressive driver implementation is desirable in order to export data written by Syndicate as first-class records in the underlying storage system. We plan to implement replication drivers for the following storage systems: (1) Local storage (including existing distributed filesystems like NFS and GlusterFS), (2) iRODS, (3) Amazon S3, (4) Dropbox, (5) Box, (6) Google Drive, and (7) Amazon Glacier.
- **Application Drivers –** Application drivers connect Syndicate to existing application processes and toolsets. They can be tailored to the application's expectations without altering the behavior of the rest of the system. We plan to implement application drivers for the following application frameworks: (1) FUSE filesystem, (2) RESTful HTTP server, (3) Hadoop back-end, (4) C++ library, (5) Python library, and (6) Key/value store.

In addition to supporting SG driver modules, Syndicate also supports authentication plugins that make it possible for Syndicate to link to an existing user database, which in turn provides a means for site admins to grant users access to volumes using single sign-on. Currently, Syndicate is capable of authenticating users via any user database that speaks the OpenID 2.0 protocol. Linking against user databases that do not support OpenID 2.0 will entail either creating an OpenID proxy to the database, or extending Syndicate to communicate via additional protocols (such as SAML). Our first target is to implement an authentication plugin for InCommon.

5.3 Evolve and Enhance the Core

Evolving Syndicate to be a better storage substrate is an important aspect of this project. Our general strategy is to gain experience by supporting real users, and then (1) distill the common, general requirements we discover into Syndicate's core architecture, and (2) identify and factor out domain-specific requirements into reusable gateway and driver implementations. This approach lets us avoid the trap of constructing domain stovepipes—Syndicate remains domain agnostic, but supports our users' specific requirements.

Our primary intellectual contribution is to identify and codify the common abstractions exposed by seemingly different usage scenarios, but in a way that best leverages existing commodity components instead of re-inventing the wheel. The following identifies general issues we expect to address. The list should be interpreted as representative of the sort of challenges that are likely to arise, prioritized according to our current understanding. The actual roadmap will be governed by feedback from the user community.

- **Live Data Analytics –** Since Syndicate acts as the gatekeeper between commodity infrastructure and on-site applications, it is well positioned to monitor all application-level storage requests. We plan to leverage this fact to build a live data analytics platform to help users trace in and replay workloads, to help the pilot team monitor and test the infrastructure's systemic behavior, and to document usage for the purpose of evaluating demand and success.
- **OS-Level Bottlenecks –** Legacy programs expect to access Syndicate via the filesystem, which potentially introduces a bandwidth bottleneck (e.g., FUSE). We will address this with a combination

of (1) researching and building a high-performance filesystem framework, and (2) instrumenting client programs to access Syndicate via a client library instead of a filesystem, without modification.

- **Driver Pipeline –** Syndicate currently permits only one driver per gateway, requiring us to address all storage-level concerns in a single module. This works for simple storage concerns, but hinders code reuse. We will address this by extending Syndicate to support "driver pipelines" (analogous to UNIX pipelines), wherein the next driver takes the output from the previous as its input. This will allow us to implement complex storage-level behaviors by composing building block drivers.
- **Cache Configuration –** It is not clear what an organization's optimal caching policy looks like, since it largely depends on which workloads run and how they interact with one another. However, the availability of the storage workflow analytics platform we plan to build on top of Syndicate will help us answer this question. Moreover, a dedicated CDN under the control of a research consortium makes it possible to automatically reconfigure itself to maximize cache hits for the current workload. We hope this allows us to automate the evaluation and evolution of domain-specific caching policies.
- **Regulatory Compliance –** We plan to explore using Syndicate to transform commodity cloud infrastructure into standards-compliant storage, suitable for handling sensitive data whose access is governed by regulatory regimes. For example, we plan to create Syndicate storage drivers that will allow scientists to use commodity cloud storage for hosting data governed by HIPAA.
- **Data Migration –** Because Syndicate can leverage multiple back-end storage services, we plan to explore using Syndicate to migrate data seamlessly between services to find the cost-performance-availability sweet spot for on-site workloads. The Syndicate storage drivers for this task would monitor the price and access performance of each back-end, as well as front-end data requests, and formulate and execute cost-effective data replication strategies subject to on-site requirements.
- **Advanced Congestion Control –** To make judicious use of space in commodity caches, Syndicate breaks down requests into fixed-sized HTTP requests and issues them in parallel. However, this circumvents the congestion control algorithm in TCP, thereby taking more than its fair share of network bandwidth. We address this by performing congestion control in the application layer within Syndicate, but systematic tests of its efficacy are still needed.

In addition to these specific optimizations, extensions, and features, we will also pursue the general evolution of the Syndicate core as the usage scenarios become more diverse. This includes:

- **Everything is a Data Source –** We want to minimize the operational overhead of importing read-only data from domain-specific data sources into Syndicate volumes. To do so, we may need to generalize the driver abstraction, making it easy and safe to link Syndicate directly to the data sources themselves—e.g., the computers and lab equipment deep in the site's network that directly generate experimental data.
- **Everything is a Data Sink –** We also want to minimize the operational overhead of incorporating data into application workflows. Syndicate preserves a common interface over all data sinks to preserve application compatibility, but it might need to be extended to allow an individual sink to expose workflow-specific extensions. For example, this may be necessary to export data from Syndicate to other data management systems.
- **Continuous Integration –** Because Syndicate will be deployed to address a wide variety of storage needs, we will be in continuous contact with our user community to look for common, emergent storage requirements, and continuously refactor the Syndicate codebase to incorporate them without losing generality. This will lower the barrier to entry for subsequent users, and reduce operational costs for existing users.

Bibliography

[1] Kyle Chard, Simon Caton, Omer Rana, and Daniel S. Katz, "A Social Content Delivery Network for Scientific Cooperation: Vision, Design, and Architecture," in *DataCloud 2012*, 2012.

[2] GlusterFS. [Online]. www.gluster.org

[3] Jakob Blomer, Predrag Buncic, and Rene Meusel. (2013, March) The CernVM File System. [Online]. http://cernvm.cern.ch/portal/sites/cernvm.cern.ch/files/cvmfstech-2.1-4.pdf

[4] Stephen A. Goff et al., "The iPlant collaborative: cyberinfrastructure for plant biology," *Frontiers in plant science*, vol. 2, 2011.

[5] iRODS. (2014, Nov) iRODS (Integrated Rule-Oriented Data System). [Online]. http://irods.org/wp-content/uploads/2012/04/iRODS-Overview-November-2014.pdf

[6] Todd Tannenbaum, Derek Wright, Karen Miller, and Miron Livny, "Condor - A Distributed Job Scheduler," in *Beowulf Cluster Computing with Linux*, Thomas Sterling, Ed.: The MIT Press, 2002.

[7] Filesystems in Userspace. [Online]. http://fuse.sourceforge.net

[8] M-Lab. [Online]. http://www.measurementlab.net/

[9] M-Lab Dataset - Google BigQuery - Google Cloud Platform. [Online]. https://cloud.google.com/bigquery/docs/dataset-mlab

[10] Metagenomics MG-RAST. [Online]. http://metagenomics.anl.gov/

[11] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler, "The Hadoop Distributed File System," in *IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, 2010.

[12] Google App Engine - cloud.google.com. [Online]. http://cloud.google.com/appengine/

[13] Shark: SQL and Rich Analytics at Scale | AMPLab - UC Berkeley. [Online]. https://amplab.cs.berkeley.edu/projects/shark-making-apache-hive-run-at-interactive-speeds/

[14] David Mazieres, Michael Kaminsky, M. Frans Kaashoek, and Emmitt Witchel, "Separating Key Management from File System Security," in *17th Symposium on Operating Systems Principles (SOSP)*, 1999.

[15] John H. Howard, "An Overview of the Andrew File System," in *USENIX Winter Technical Conference '88*, 1988.

[16] Jeremy Stribling et al., "Flexible, Wide-Area Storage for Distributed Systems with WheelFS," in *6th NSDI*, 2009.

[17] Marc Shapiro, Nuno Preguicca, Carlos Baquero, and Marek Zawirski, "Conflict-free replicated data types," in *13th International Conference on Stabilization, Safety, and Security of Distributed Systems*, 2011.

[18] Francisco J. Torres-Rojas, Mustaque Ahamad, and Michel Raynal, "Timed Consistency for Shared Distributed Objects," in *PODC '99*, 1999.

[19] KyoungSoo Park and Vivek Pai, "Scale and Performance in the CoBlitz Large-File Distribution Service," in *Proc. 3rd Symposium on Network Design and Implementation (NSDI)*, 2006.

[20] James Gwertzman and Margo Seltzer, "World-Wide Web Cache Consistency," in *1996 USENIX Technical Conference*, 1996.

[21] David Recordon and Drummond Reed, "OpenID 2.0: a platform for user-centric identity management," in *Second ACM Workshop on Digital Identity Management*, 2006.

[22] Amazon S3, Cloud Computing Storage for Files, Images, Videos. [Online].

http://aws.amazon.com/s3/

[23] DropBox. [Online]. http://www.dropbox.com

[24] Box | Secure content-sharing that users and IT love and adopt. [Online]. https://www.box.com/

[25] Amazon Glacier. [Online]. https://aws.amazon.com/glacier/

[26] Google Drive. [Online]. https://www.google.com/drive/

[27] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403-10, Oct 1990.

[28] GenBank Flatfile dataset. [Online]. ftp://ftp.ncbi.nih.gov/genbank

[29] GenBank ASN.1 dataset. [Online]. ftp://ftp.ncbi.nih.gov/ncbi-asn1

[30] GenBank Statistics. [Online]. http://www.ncbi.nlm.nih.gov/genbank/statistics

[31] Brian Tierney et al., "perfSONAR: Instantiating a Global Network Measurement Framework," in *4th Workshop on Real Overlays and Distributed Systems*, 2009.

[32] Andy Bavier et al., "Operating System Support for Planetary-Scale Network Services," in *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI 2004)*, 2004.

[33] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," in *Proceedings of HotNets - I*, 2002.

[34] Larry Peterson, Andy Bavier, Marc Fiuczynski, and Steve Muir, "Experiences Building PlanetLab," in *Proc. 7th Operating System Design and Implementation (OSDI)*, 2006.

[35] PlanetLab Bibliography. [Online]. https://www.planet-lab.org/biblio

[36] PlanetLab Courseware. [Online]. https://www.planet-lab.org/courseware

[37] GENI. [Online]. https://www.geni.net/

[38] OneLab. [Online]. https://www.onelab.eu/

[39] Akihiro Nakao, Ryota Ozaki, and Yuji Nishida, "CoreLab: An Emerging Network Testbed Employing Hosted Virtual Machine Monitor," in *ACM ROADS 2008*, 2008.

[40] G-Lab. [Online]. http://german-lab.de/

[41] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford, "In vini veritas: realistic and controlled network experimentatio," in *Proceedings of SIGCOMM '06*, 2006.

[42] Larry Peterson, Andy Bavier, and Sapan Bhatia, VICCI: A Programmable Cloud-Computing Research Testbed, September 2011.