Building Access Oblivious Storage Cloud for Enterprise

Hyunseok Chang Murali Kodialam T. V. Lakshman Sarit Mukherjee Limin Wang
Bell Laboratories, Alcatel-Lucent
Murray Hill, NJ, USA

Abstract

An enterprise uses VPNs, leased from a service provider, to interconnect multiple sites that are geographically apart. The service providers, as they start providing cloud-based services, are finding themselves well-positioned to providing storage services in the cloud for an enterprise, and make the service accessible through the existing VPN connections. Enterprise users, however, are used to fast, ubiquitous and guaranteed access to the storage from any enterprise location. This is achieved by having network attached storage (NAS) connected to the enterprise network. In order to maintain the same level of service, when the enterprise storage is moved into the cloud, the service provider must ensure that the storage is accessible from all the enterprise locations as if it is connected to the enterprise network itself, regardless of the actual user or the file. In this paper, we present a system that enables cloud storage service with guaranteed performance from all published access locations of an enterprise. Knowing only the limits on users access rates or their access bandwidth limitations, we develop an access oblivious storage provisioning and placement strategy. Our system uses a combination of chunking, data replication and intelligent data placement to guarantee performance to accessing the storage in an access independent manner without significant over-provisioning.

1 Introduction

A typical enterprise of today manages its computing, storage and networking needs, i.e., IT services, in-house. It procures all the computing, storage and networking that its users require, and maintains them within the enterprise in one or a few geographically distant locations. All the enterprise locations are usually interconnected using VPN connections leased from a service provider. This model allows the enterprise to make the resources available to any user from any location using the enterprise network, and gives the enterprise complete control on the resources. This model, however, is both capital and operational expense intensive. Moreover, since the demand for IT resources is not known a priori, appropriate sizing of the resources is extremely difficult. To make the matter even worse, the demand for resources could vary widely at different time of the day or days of the month. Cloud computing, on the other hand, provides a flexible model in allocating computing and storage resources so that resources can be added or removed depending on the demand.

As mentioned before, an enterprise is used to the VPN outsourcing model to interconnect multiple locations over a service provider's geographically distributed network. The service providers' desire to expand their footprint into the compute and storage cloud business fits very well with enterprise's logical choice for outsourcing its compute and storage resources into the cloud. In such a scenario, an enterprise may maintain a skeletal enterprise IT services at its location(s), and outsource the bulk, along with the VPN, into the cloud maintained by the service provider. Access to the cloud-based compute and storage resources is enabled by the leased VPN connections into the service provider's distributed cloud. The biggest challenge for the enterprise IT services in this case is how to leverage the service provider's cloud services to satisfy the computing, storage and networking needs of its users. Several aspects of user expectations from IT services must be satisfied when virtual resources in the cloud become part of the enterprise:

- **Isolation:** The enterprise's virtual resources in the cloud must be isolated from the other users of the cloud, and must be accessible by the users of the enterprise in question only.
- **Location independence:** The enterprise users must be able to connect to the virtual resources in the cloud from any enterprise location.
- **Seamlessness:** An enterprise user must not see any difference between accessing an in-house resource vs. accessing one in the cloud, and access resources as if they are connected to the enterprise network.

In this paper we focus on using virtual storage resources in the service provider cloud to satisfy an enterprise's storage needs. Traditionally an enterprise connects NAS equipment to its LAN to enable a shared file-based storage for its users. The enterprise LAN provides enough bandwidth to each user to ensure fast file access, which emulates the experience of a directly connected storage. When an enterprise outsources the storage into the cloud, it must ensure the same level of isolation, location independent and seamless access to the storage. Different techniques have been proposed [4, 8] to extend an enterprise LAN into the cloud with all the security and isolation of a LAN. In this paper we assume that storage isolation is provided by hooking up the (virtual) storage device for the enterprise to its emulated LAN in the cloud, and propose a solution to ensure location independent and seamless access to the storage.

1.1 Our Contribution

We present a novel architecture and a prototype system for storage provisioning and placement in a service provider's cloud that augments service provider's VPN service with enterprise grade storage services. It enables a bandwidth guaranteed, seamless and location independent file system in the provider's cloud. By using a combination of data chunking, data replication and intelligent data placement, it handles the access uncertainty in file system access and thereby provides seamlessness and location independence. We develop a linear programming formulation and a primal-dual algorithm for the access oblivious placement of enterprise data that achieves the placement without significant over-provisioning of the cloud resources. To the best of our knowledge, this is the first approach for designing a cloud storage that provides guaranteed performance without prior knowledge of file access patterns. The proposed architecture has several advantages over traditional distributed file systems. First, its access obliviousness allows data placement to not change with dynamic file access patterns. New users can join easily at any location and new data can be ingested into the cloud so long as the bandwidth demands remain within the VPN service limits. The static data placement also translates into a significantly reduced operating cost of running a storage cloud, by eliminating the need for monitoring and constantly shuffling the data in the cloud to meet performance requirements with changing access patterns. Finally, it is easy for the storage cloud to determine if an enterprise's request can be accepted into the system with the required performance guarantees, which makes admission control decisions straightforward.

2 Enterprise Storage Cloud Architecture

Fig. 1 shows the overall architecture of a distributed storage cloud augmented to the VPN service enabled by the service provider. At the edge of the cloud there are edge

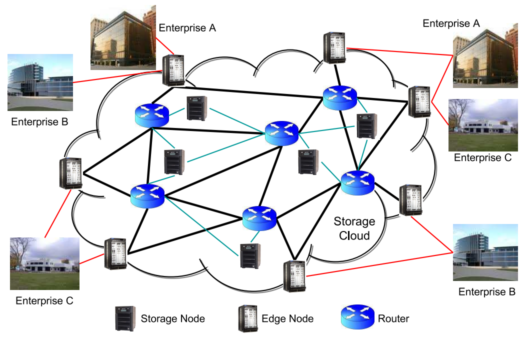

Figure 1: Architecture of the storage cloud.

nodes through which different sites of the enterprise connect to avail the VPN service. At the time of leasing the VPN service, the enterprise specifies the bandwidth for the VPN connection, which is also used to access the storage in the cloud. The architectural details of the VPN service is beyond the scope of this paper.

The distributed storage cloud is an interconnection of geographically dispersed storage nodes. It shares the same edge nodes as the VPN service. The enterprise accesses the storage using the pre-negotiated VPN connections from its different sites. Thus, the bandwidth to access storage is limited by the negotiated bandwidth of a VPN connection. The service provider provisions some of the storage nodes to create virtual storage for the sole use of the enterprise. From then onwards the storage cloud guarantees that any user of the enterprise is able to access any data of the enterprise from any of the enterprise sites using the corresponding VPN connections. This qualifies the service as an enterprise grade service.

The crux of the problem is to select a set of storage nodes from a distributed cloud, for housing the enterprise data in such a way that provides guaranteed access to the data without a priori knowledge of user access pattern, access location and actual accessed files. One solution for the service provider is to track the access pattern of the data and move data around the storage nodes so that access can be guaranteed. However, this is not a very practical solution as it involves movement of a large amount of data in real-time within the storage cloud. Another option would be to significantly over-provision the network so that there is no bottleneck in the delivery capacity. This is of course not an attractive option when revenue-per-bit is amongst the lowest for the service provider network. Ideally, a storage cloud must have the following characteristics:

- The delivery network must scale with the number of users and the amount of data.
- There should be no bottleneck at the storage nodes independent of the files being accessed.
- There should be no bottlenecks at the network links independent of the accessed files.

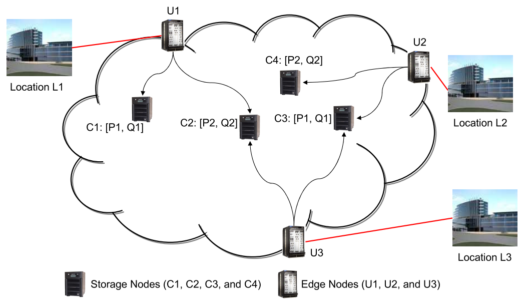

Figure 2: An example of enterprise storage provisioning and file access.

Meeting these requirements essentially will give each user the perception that all files are stored on a local server. So the best policy is to provision and place the storage in a way that is oblivious to the access pattern of the users. This type of service contract has several advantages for the enterprise. First, the service contract effectively eliminates the access uncertainty problem for the enterprise and mitigates the location uncertainty problem significantly. The service model also makes it easy for the enterprise to renegotiate, explicitly or implicitly, its contract by either augmenting or cutting back the access rate at the different edge nodes.

2.1 Storage Provisioning and File Access

If there is only one copy of every file at some single storage node, and if this file is popular, then the network close to this storage node usually becomes a bottleneck for data delivery even if there is enough server capacity at the storage node for serving out the data. Two different mechanisms can be employed to avoid these congestion bottlenecks in the network.

- *Data Replication:* Data replication is the most commonly used method to distributing the network load across various links and thereby reducing the load on any individual link.[1] Multiple copies of each file are stored at different storage nodes and a request for the file is served from any one of the nodes that has a copy of the file. A commonly used strategy is to pick the closest storage node that has the file.
- *Data Chunking:* In data chunking each file is split into several smaller chunks which may be stored at different storage nodes. When a file is requested, all the chunks are simultaneously downloaded to the user, assuming their locations are known. Chunking enables the load to be distributed across the network.

With data replication and chunking, we now focus on how an enterprise with multiple sites accesses the outsourced cloud storage using the VPN connection. We depict an example scenario in Fig. 2 where an enterprise subscribes to the VPN and the storage service at three locations, L1, L2 and L3. Each site connects to the cloud at the close-by edge nodes, U1, U2, and U3, respectively, via the VPN connection. Assume that storage for the enterprise is provisioned into four storage nodes, C1, C2, C3 and C4. All data belonging to the enterprise is split into two chunks, and all chunks are replicated twice. In the figure, we show how two files P and Q can be stored in the storage nodes. When a user from a site L1 accesses either file, the request goes through the VPN connection to the corresponding edge node U1. It is the edge node that acts as a gateway into the file system and hides all the provisioning details from the users. In this case, U1 maintains a table per enterprise that directs it to get two chunks, one from C1 and another from C2 for any file accessed by the enterprise in question from site L1. Similarly U2 and U3 get the chunks from (C3, C4) and (C3, C2), respectively. To keep the cloud resource usage limited, we assume that an edge node selects the closest storage nodes to retrieve the chunks. However, it is easily extensible to other strategies like tunneling into a specific storage node which may be farther away. After an edge node retrieves the chunks of a file, it delivers them to the user of the enterprise. Typically the edge node runs a network file system proxy that the user mounts. To make the architecture scalable, an edge node does not partake in any file system operations, other than proxying for the storage nodes in the cloud. The storage nodes are responsible for all file system operations including maintaining multiple copies consistent.

Note some of the features of the example storage provisioning. The storage nodes that an edge node accesses for a particular enterprise is always fixed regardless of the file being accessed. Moreover, if we keep the chunking ratio uniform across the files, i.e., size(P1)/size(P2) = size(Q1)/size(Q2) = r (say), then the ratio of traffic between the edge node and storage node paths remains at r, i.e., traffic(U1, C1)/traffic(U1, C2) = r. This makes the storage provisioning oblivious to the actual file per se. In the following section we describe our access oblivious provisioning methodology in more detail.

3 Access Oblivious Storage Provisioning and Placement

We model the storage cloud of Fig. 1 to consist of a set of storage nodes C, a set of edge nodes U and a set of E links interconnecting them. Enterprise data, i.e., files etc., is stored and delivered from the storage nodes whose locations are fixed and known. While both data uploads and downloads (i.e., write and read operations

[1]While data replication can also be used for handling storage node redundancy, we do not consider storage fault tolerance in this paper.

into the file system, respectively) are handled by the storage cloud, for simplicity we describe the case of download in the rest of the paper. Note that upload can be handled in a similar fashion.

The enterprise connects to the cloud at one of the edge nodes. All the data in the cloud flows from the nodes in C to the nodes in U and then to the enterprise site. We assume that the maximum rate at which data can be downloaded at the edge node u is upper bounded. Typically, this upper bound is the capacity of the VPN connecting the enterprise site into the cloud at edge node u. Let D_u be the maximum permitted download rate at u. We use F to represent the set of files currently present in the cloud that can be accessed by an enterprise user. The set of files in the system is, therefore, dynamic, and F just represents the current snapshot of the set of files. A file is stored in one or more storage nodes belonging to C. Upon a request from the user, the file is delivered to the user at an edge node using the shortest path routing within the cloud. The currently available capacity of a link e is denoted as $c(e)$ which is the original link capacity reduced by the bandwidth reservations made on the link for all accepted enterprise customers. An enterprise is admitted by the cloud service provider only if there is sufficient bandwidth available to process any access pattern for the files. Once an enterprise is admitted, the remaining bandwidth on the network links are updated. In order to compute the bandwidth requirement for downloading the file, we assume that there is some quality of service agreement between the enterprise and the cloud storage provider where the cloud storage provider guarantees the minimum rate at which a file will be downloaded from the storage nodes to the edge node where the enterprise site's VPN connects to. Let R^f denote the download rate for file f. Let $N_u^f(t)$ represent the number of requests for file f being transmitted to edge node u at time t. The total bandwidth requirement at u at time t is then $\sum_{f\in F} N_u^f(t)R^f$. Since the maximum download rate at u is D_u, we know that

$$\sum_{f\in F} N_u^f(t)R^f \le D_u \ \ \forall u,t. \tag{1}$$

As stated earlier, chunking and replication are two mechanisms that can be used for efficient dissemination of data in network. We use both these mechanisms in order to devise an access oblivious scheme. We assume that each file in the network is chunked into at most p chunks, and that each of these chunks of each file is replicated r times. We now consider the problem of designing an access independent cloud based storage service mechanism. The decisions that have to be made are: (1) the size of each chunk, (2) the storage nodes where each replicated chunk is placed, and (3) the storage node from which each chunk of a requested file is to be downloaded by a given enterprise edge node. We assume that all files are split into the same number of chunks. Let β_k represent the fraction of each file that is in chunk k. We impose the constraint $\sum_k \beta_k = 1$ to ensure that the entire file is chunked. In addition we impose the constraint $\beta_k \ge \frac{1}{p}$ in order to ensure that each file is split into at most p chunks. (This also ensures that none of the chunks is too small). Note that chunk k is replicated r times. We place replica j of chunk k of all files in the same location. The decision variables are the size of the chunks and where each replica of each chunk is placed. Since there are $|C|$ storage nodes and each chunk is replicated r times, there are $q = \binom{|C|}{r}$ replication choices. We call each of these sets a *replication group*. We use G_j to represent the set of r nodes in replication group j. If chunk k is stored in replication group G_j, then *each* node $c \in G_j$ stores chunk k of *every* file. In order to keep notation simple, we assume that the chunk id and the replication group id are the same. In other words, we assume that chunk j is stored in replication group G_j. When an edge node u downloads a fraction β_j of a file from replication group j we assume that the download is done from some node $c \in G_j$ that is closest to u. In other words, node u downloads a fraction β_j from node $c \in G_j$ such that $|SP(c,u)| \le |SP(c',u)|$ for all $c' \in G_j$, where $SP(.,.)$ denotes the shortest path between two nodes. We use $SP(G_j,u)$ to denote the set of links in $SP(c,u)$ where c is closest node in G_j to node u. This assumption of download from the closest replicate is done purely for convenience of implementation. The solution technique can be extended easily to solve for the case where the file can be downloaded from multiple nodes in the same replication group. Since each node downloads a fraction β_j of every file from replication group j, the total flow rate from replication group j to a node u at time t is $\sum_{f\in F} \beta_j N_u^f(t)R^f$ where $N_u^f(t)$ represents the number of copies of file f being transmitted to node u at time t. From Equation [1] we can write:

$$\begin{aligned}\sum_{f\in F} \beta_j N_u^f(t)R^f &= \beta_j \sum_{f\in F} N_u^f(t)R^f \\ &\le \beta_j D_u \ \ \forall u,t.\end{aligned}$$

Therefore, the amount of capacity to be provisioned from replication group j to node u is upper bounded by $\beta_j D_u$. Let $H(e)$ denote the set of (replication group, edge node) pairs that use link e to download data. In other words, $H(e) = \{(G_j,u) : e \in SP(G_j,u)\}$. The maximum flow on link e will then be $\sum_{(G_j,u)\in H(e)} \beta_j D_u$. We can now formulate the problem of determining the the optimum chunking and replication as a linear programming problem. The objective of the linear programming problem is to determine how the files are chunked and where these chunks are replicated in order to minimize the maximum link utilization represented by λ.

The linear programming problem to determine the optimal chunking parameters is the following:

$$
\begin{aligned}
&\min \lambda \\
\sum_{(G_j,e)\in H(e)} \beta_j D_u &\leq \lambda\, c(e) \;\; \forall e \in E. \\
\sum_j \beta_j &= 1 \\
\beta_j &\geq \frac{1}{p} \;\; \forall j.
\end{aligned}
$$

If the optimal solution value to this problem is $\lambda \leq 1$, then the enterprise can be admitted with the access oblivious service guarantee. If $\lambda > 1$, then the enterprise can be admitted if the maximum download rate is scaled down to D_u/λ at each enterprise edge node u. If there are storage amount constraints at the any of the storage nodes, it can be easily incorporated into the linear programming formulation. It is also possible to solve a reformulation of the above linear programming problem using a simple primal dual algorithm. This is omitted due to lack of space.

4 Prototype Implementation and Experimentation

We have implemented a prototype enterprise storage cloud with guaranteed performance using our access oblivious provisioning and placement solution. Our prototype system takes enterprise's service level agreement (SLA) inputs when admitting a new customer. These SLAs include enterprise's storage capacity requirement, VPN edge locations and more importantly, expected guaranteed access rates at different edge sites. Next, our access oblivious provisioning algorithm devises a placement plan for the enterprise based on the SLAs. This plan then gets executed in our storage cloud, and access to the cloud data is made available to the enterprise through proxies running on the edge nodes. Currently, these proxies expose an NFS interface.

Our storage cloud prototype, shown in Fig. 3 is built on top of Ceph [7]. Ceph is an open source, scalable, high-performance object-based distributed network file system. In Ceph, files are striped onto multiple objects that are independently and synchronously stored and replicated to Object Storage Devices (OSDs). It decouples data and meta data management through deploying separate (distributed) metadata server(s) (MDS). Ceph handles data migration, replication, failure detection and recovery natively.

We choose Ceph due to its open source nature and its support in the Linux community. Ceph's separation of meta data and data operations makes it easier to employ customized data placement strategies. We use Ceph

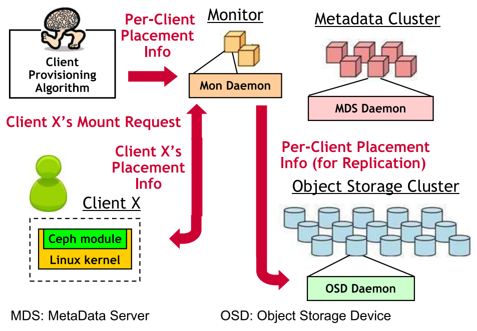

Figure 3: Prototype implementation.

OSDs to hold user data, and leverage MDS to implement our access oblivious provisioning and placement algorithm by modifying Ceph's data placement logic. We make the location of a given object uniquely determined by (1) the owner of the object, and (2) the gateway location, i.e., the edge node proxy, at which the owner is mounting her remote share. All the modifications are built into the Linux kernel supporting Ceph. We modify some of the replication strategies in Ceph to create replication group for each enterprise. Our edge node proxies then export the enterprise's data in OSDs as an NFS share. A full-blown description of the implementation details is not included due to space restriction.

The whole storage cloud connecting multiple storage nodes and edge nodes is deployed in our lab data center. Dedicated laptops are used as enterprise customers, and are connected through VPN tunnels to designated edge nodes based on enterprise SLAs specified at customer admission time. Enterprise user requests are then made from these laptops. There is also a web based control interface for admitting new enterprise customers and monitoring storage cloud in real time. The system has been fully functioning, and has been used in experimental cases such as video streaming from enterprise sites, providing needed performance guarantees.

Besides prototype implementation, we have also conducted a simulation experiment of our access-oblivious storage provisioning scheme to compare its bandwidth resource usage against that of *access-aware* counterparts. In the latter, information on file access patterns (e.g., what file is accessed by which edge nodes, and when) is explicitly taken into account to derive the *time-dependent*, *per-file* chunking and replication strategies. In our experiments, we consider two such schemes: (1) bandwidth-aware (BA) provisioning and (2) proximity-aware (PA) provisioning. The BA provisioning determines the optimal chunking and replication per-file such that the maximum link utilization remains minimized with changing access patterns. The PA scheme, on the other hand, splits and places files as close as possible (in terms of network hops) to the edge nodes accessing them, without considering link capacity constraints.

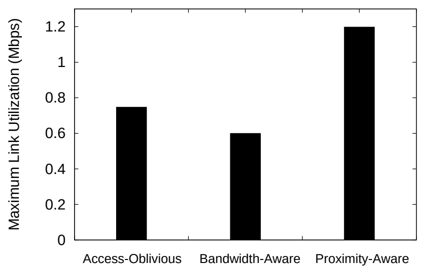

Figure 4: Maximum link utilization.

We test three provisioning scenarios driven by access-oblivious, BA, and PA schemes, respectively. In all scenarios, the same number of customers are admitted to the system, who then generate the same file access traffic on storage network.[2] We compare three schemes in terms of the maximum link utilization of the network. Fig. 4 shows that the access-oblivious provisioning results in 25% higher maximum link utilization compared to the BA provisioning, which implies that the access-oblivious provisioning could be achieved without significant over-provisioning. Also, note that 25% savings on bandwidth resources with the BA scheme is actually achieved with added complexity of monitoring instantaneous file access patterns and dynamically relocating individual file chunks on the fly, and without considering additional bandwidth required for file relocation. The PA provisioning exhibits significantly poorer performance than the other schemes. This comes as no surprise because, when files are placed only in network proximity in mind, well connected storage nodes could host most requested files, and so the adjacent links of such nodes would become heavily congested.

5 Related Work

There are multiple commercial vendors in the online storage market. For example, major cloud providers such as Amazon, Google and Rackspace offer storage service as part of their Infrastructure as a Service offerings. Online file hosting providers deliver various value-added services such as online backup (e.g., Dropbox) and sharing (e.g., Slideshare, SkyDrive) on top of available cloud storage infrastructure, through user-friendly interfaces. Storage solution providers such as NetApp, EMC, Hitachi Data Systems, IBM target enterprise customers with proprietary NAS devices that enable rapid provisioning of private enterprise storage systems. In all these solutions, however, customers have no precise control over how the data is stored in the cloud, and what level of quality of service is expected.

[2]Our simulation is performed using GTNetS, a discrete-event network simulator which is driven by YouTube access traces [9] on top of RocketFuel's router-level ISP maps with uniform link capacity.

Several researchers proposed new distributed network file systems optimized for scalability and access performance. The Swift [2], Zebra [5], PVFS [6], Ceph [7] file systems leverage the mechanism of striping files across different storage servers, thereby distributing the load and achieving high access throughput. The Google File System [3] was designed to meet the demand of a typical data center environment where data-intensive applications run off of a large pool of commodity hardware. While these file systems have extensive architectural support for file chunking and replication, the focus is not on designing effective file chunking and placement strategies. Our scheme addresses this latter issue. Cloud storage services can migrate client data in response to dynamic access patterns and client locations, using an iterative optimization algorithm such as [1]. However, they typically need heavy duty data collection and computation, and thus cannot sustain fine granular time scales.

6 Conclusions

We presented the design of an enterprise grade storage cloud that can augment a service provider's VPN service offer to an enterprise. The storage cloud offers a performance guarantee for access to data stored in the cloud that is independent of the data being accessed and the location it is accessed from, and gives users the perception that their data of choice is stored locally. We developed algorithms for the design of such access-oblivious guaranteed performance storage cloud. We outlined a linear programming formulation for deciding how the data is placed across different nodes, and described a prototype system that implements the storage architecture in the Linux kernel on top of Ceph.

References

[1] AGARWAL, S., ET AL. Volley: Automated Data Placement for Geo-Distributed Cloud Services. In *NSDI* (2010).

[2] CABRERA, L., AND LONG, D. D. E. Swift: Using Distributed Disk Striping to Provide High I/O Data Rates. *Computing Systems 4*, 4 (1991).

[3] GHEMAWAT, S., GOBIOFF, H., AND LEUNG, S.-T. The Google File System. *ACM SIGOPS Operating Systems Review 37*, 5 (2003).

[4] HAO, F., LAKSHMAN, T., MUKHERJEE, S., AND SONG, H. Secure Cloud Computing with a Virtualized Network Infrastructure. In *HotCloud* (2010).

[5] HARTMAN, J. H., AND OUSTERHOUT, J. K. The Zebra Striped Network File System. *ACM Transactions on Computer Systems 13*, 3 (1995).

[6] LATHAM, R., MILLER, N., ROSS, R., AND CARNS, P. A Next-Generation Parallel File System for Linux Clusters. *LinuxWorld* (Jan 2004).

[7] WEIL, S. A., ET AL. Ceph: A Scalable, High-Performance Distributed File System. In *OSDI* (2006).

[8] WOOD, T., SHENOY, P., RAMAKRISHNAN, K., AND MERWE, J. The Case for Enterprise-Ready Virtual Private Clouds. In *HotCloud* (2009).

[9] ZINK, M., SUH, K., GU, Y., AND KUROSE, J. Characteristics of YouTube network traffic at a campus network - Measurements, models, and implication. *Computer Networks 53*, 4 (2009).