

AmazingStore: Available, Low-cost Online Storage Service Using Cloudlets *

Zhi Yang*, Ben Y. Zhao†, Yuanjian Xing*, Song Ding*, Feng Xiao* and Yafei Dai*

*Computer Science Department, Peking University, Beijing, China

†Computer Science Department, U. C. Santa Barbara, CA, USA

{yangzhi, xyj, dingsong, xiaofeng, dyf}@net.pku.edu.cn, ravenben@cs.ucsb.edu

ABSTRACT

“Cloud-based” Internet services rely on the availability and reliability of managed data centers. Recent events indicate that data centers tend to create centralized points of failure, and providing resilience to large-scale faults remains a significant challenge for both providers and users of cloud infrastructures. Running data centers also incurs high hardware and network costs, particularly for storage-intensive applications such as data synchronization and backup. In this paper, we show how to improve data availability while reducing costs in storage clouds, by augmenting centralized clouds with an efficient client-side storage system. We introduce AmazingStore, a low-cost cloud storage system that provides high data availability while protecting against correlated failures. We describe our initial experiences with an already deployed prototype and outline opportunities in this modified cloud model.

1. INTRODUCTION

Always available and highly durable storage is one of the driving applications of the cloud computing model. The ability to store or backup files onto highly reliable data centers promises to greatly simplify data management at the edge, a growing challenge for users with rapidly increasing number of files. Users of these popular services can use them as online storage services (*e.g.* Amazon’s S3 and Microsoft’s SkyDrive and LiveMesh), or purely for online backup (*e.g.* Symantec’s SwapDrive and EMC’s Mozy).

There are two primary challenges facing cloud-based services, especially the storage services. First, cloud infrastructures are generally centralized for manageability and economy. Therefore, they are vulnerable to central point of failures caused by fires, power outages, natural disasters, *etc.* In fact, a quick scan of recent news shows that in the first half of 2009 alone, cloud computing providers such as Google, Amazon, and Rackspace have been hit by several major outages attributed to numerous causes, including network hardware failure [12], fires [6], network management [7], power

failure [13], and other undisclosed causes. Clearly, cloud providers have not embraced replication across distributed data centers for fault-tolerance, perhaps due to challenges of performance and data consistency [15]. A second challenge lies in the cost. For cloud storage providers, highly available storage means more storage arrays and power backups, escalating the cost of network bandwidth, data center cooling, and well-trained IT management personnel [14].

In this paper, we take the first step to address these challenges by proposing *Cloudlets*, an alternative model for building available, reliable and economically efficient cloud-based storage services. In this model, we augment the centralized cloud-based storage service with a P2P storage service built using storage resources at its users. Each peer provides a *cloudlet* of storage for a number of object replicas in the system. Data access requests are forwarded to replicas on cloudlets whenever possible. Our objective is to improve the cloud’s resilience to correlated failures while decreasing storage and bandwidth costs. Our key insight is that since the on/off availability dynamics of cloudlets are independent from failures at data centers, the peer-based storage layer complements the central cloud and maintains data availability during data center outages. In addition, available data replicas on peers can divert traffic away from the data center, thereby reducing its workload.

We instantiate the Cloudlets architecture in *AmazingStore*, a highly available, reliable, general purpose storage system. While purely peer-based storage systems have achieved limited success in the past [2], *AmazingStore* incorporates several novel insights and lessons from prior systems that make it well-suited to support today’s online storage demands. First and foremost, since *AmazingStore* stores at least a “master” replica of each data object in the data center, its peer-based storage layer does not need to guarantee perfect availability. Second, it incorporates the advantages of both centralized and P2P architecture, thus simplifying the data management while protecting the service against single-point failures. Third, *AmazingStore* uses a group availability estimator to carefully navigate the cost-availability tradeoff at its peer level, by reducing replication costs as much as possible while maintaining the desired level of data availability. Finally, *AmazingStore* leverages the fact that requests for data

*This work is supported by the National Natural Science Foundation of China grant 60873051 and National Grand Fundamental Research 973 program of China grant 2009CB320504.

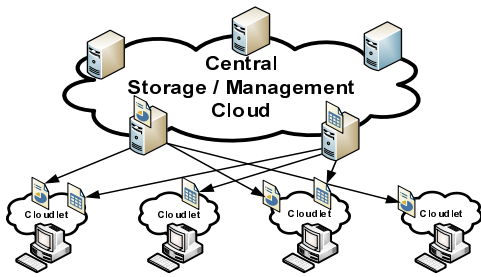


Figure 1: The Cloudlets architecture. Data is kept at a central cloud, but is replicated on distributed cloudlets for improved availability and durability.

access correlates positively with peer availability. Thus periods of high demand for data access is also the time when the most peers (and the data replicas they host) are available in the system. This means its peer level can effectively reduce the costs at the data center by allowing the number of available replicas to fluctuate with the request load.

We integrate these insights into the first prototype of AmazingStore, which has been open to users since April 3rd, 2009. As of early October 2009, AmazingStore has a growing population of over 7100 registered users and 705GB allocated storage. Given a requirement of two nines availability, AmazingStore’s clouplet peers are able to improve overall data availability from 93.22% to 99.13% and divert over 90% request traffic away from the clouplet servers, while reducing clouplet replication traffic to a fixed maximum of 2KB/s. Moreover, clouplet peers provide high data durability (an average of 0.07% files lost per month on the peer side), hence the replication costs required for maintaining high data durability can be further reduced.

In the remainder of this paper, we describe in detail the Cloudlets architecture, as well as the design and motivation behind the AmazingStore prototype. We present preliminary measurement results from our deployment of AmazingStore in Section 4 and discuss opportunities for future exploration in Section 5. Finally, we conclude the paper in Section 6.

2. CLOUDLET MODEL

At a high level, clouplet includes both a centralized cloud of servers in a data cluster (“clouplet servers”), and distributed “clouplet peers” at the network edge, as shown in Figure 1. Clouplet servers serve two purposes, to provide a highly available copy of every object, and to monitor and manage replicas stored on clouplet peers. In contrast, clouplet peers are instances of the clouplet clients running on user machines. Each peer allocates a fixed amount of local disk space for replica storage, maintains periodic heartbeats with clouplet servers, and creates or deletes object replicas according to the servers’ instructions.

This model exploits the different nature of faults in data center and P2P systems. In a storage system where each

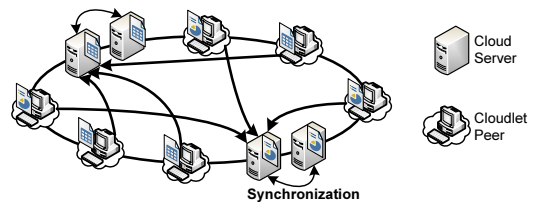


Figure 2: The AmazingStore design. Objects are organized into groups, each of which has a master node (locatable by searching for the group ID in the DHT) which monitors all replicas of objects in the group, and resolves client queries.

object is replicated in both a “small” data center and a P2P storage layer, neither has to be perfect. When an object’s P2P replicas are all offline, it will be available in the data center; on the rare occasion when the data center is offline, the P2P replicas are still available. The probability that failures coincide on both sides is low enough to guarantee high availability. And of course, this hybrid model has the potential to lower the cost for providers by turning each of its users into part of the cloud.

Notice that Clouplet is *orthogonal* to peer-assisted content distribution applications (including content distribution [4], file sharing [16] and Video-on-Demand [5]) due to different design issues. Specifically, Clouplet pertains to the reliability aspect of storage and must exhibit high availability and data persistence characteristics. Efficient replication/repair strategies are the most important design decisions when deploying a peer-assisted storage system. However, Clouplet can also adopt P2P file-sharing technology (e.g., BitTorrent protocol or DHT-based file sharing approach) to speedup the downloading of popular files.

3. AMAZINGSTORE

In this section, we briefly describe the design rationale behind AmazingStore, showing how to turn clouplet model into a practical system.

3.1 System Architecture

To form a highly available architecture, all nodes (servers and peers) are organized in a structured P2P overlay (DHT). Objects are organized into object “groups”. Each group has its own group ID and a “master node” that manages replicas of all objects within the group. The group ID (GID) maps to its “master” node on a clouplet server, as shown in Figure 2. A group master maintains pointers to all nodes with replicas of objects in the group, and monitors their liveness using heartbeats. A client seeking to read an object has its two-part ID. It first locates the object’s group master using DHT routing on its GID, then queries the master with the object’s Object ID (OID). If a replica on a clouplet peer is available, the master directs the client to that peer. Otherwise, the master returns a reference to a server replica. In this way, the

system congregates the master roles to the cloudlet servers, enabling *insensitivity* to peer dynamics.

For fault-tolerance, each master has a backup node with which it periodically synchronizes all metadata (references to replicas). The master and backup both have system-chosen nodeIDs based on the groups they manage. The backup's ID is assigned so that if the master fails, DHT routing maps any requests for the group directly to the backup. *In the worst-case scenario, e.g., the centralized cloud goes down due to correlated failures, both nodes fail, and a cloudlet peer P whose nodeID is next closest to GID receives both requests for the group and heartbeats from nodes with object replicas. Within one heartbeat period, P will receive data on the location of all replicas for the group, and can service any read requests like the original master. When the cloudlet server returns, peers can automatically switch back to the original master by periodically looking up the GID.*

3.2 Availability Maintenance

To efficiently achieve the desired level of data availability, we must carefully determine the degree of redundancy used to tolerate availability transients and how quickly the system reacts to peer departures, a significant challenge in highly dynamic environment like AmazingStore.

Recall that in cloudlet, each object is replicated on both cloudlet servers and peers. Let μ_S and μ_P represent mean server availability and peer availability respectively. Given a target level of availability A (where A represents the probability a file can be accessed at any time), we can calculate the number of required replicas in the peer layer, c , as follows:

$$A = 1 - (1 - \mu_S)(1 - \mu_P)^c \quad (1)$$

Notice that replicas placed at cloudlet servers would suffer correlated failures, so we use $1 - \mu_S$ as a conservative prediction of unavailability experienced in the server layer. Since peers's online behaviors are independent [1], we predict unavailability experienced in the peer level as $(1 - \mu_P)^c$. Solving for c ,

$$c = \frac{\log(1 - A) - \log(1 - \mu_S)}{\log(1 - \mu_P)} \quad (2)$$

However, maintaining the c replicas over longer periods is the real challenge, which is complicated by the fact that peer failures can be either transient or permanent. Notice an object with sufficient replicas (*e.g.* $\geq c$) can tolerate some transient failures without sacrificing data availability. Thus, it is preferable to accurately distinguish failures, and only pay the cost of object recreation under permanent failures. Carbonite [3] tried to address this challenge by reintegrating returning replicas, but the approach is ineffective in dynamic environments with low peer availability.

Having recognized that permanent failure detection is difficult on a per-peer basis, we improve detection accuracy by making replica liveness detections across groups of replicas (*e.g.* all peers hosting replicas of the same object), balancing

false positives for some peers against false negatives for others. In particular, we use a probabilistic method to estimate the number of *alive replicas* (*e.g.* replicas residing on online peers or peers experiencing transient failures). The first step of this method is to obtain the failure probability $F(d)$ that a peer has permanently failed given an observed failure of d time units, which can be extracted from historical traces of peer failure events (details can be found in [17]). Next, we aggregate failure probabilities across peers in the replica group and apply the maximum a posteriori (MAP) estimation rule to derive an estimate. For instance, let x be the number of remaining replicas at the time of estimation. Given $F(d_i)$ ($1 \leq i \leq n$) for peers in the group, we can compute the probability of $x = k$ using the following approximation (with accuracy/complexity tradeoff):

$$P(x = k) = \begin{cases} 0 & k < n_{up} \\ \binom{n-n_{up}}{k-n_{up}} \bar{F}^{n-k} (1 - \bar{F})^{k-n_{up}} & n_{up} \leq k \leq n \end{cases} \quad (3)$$

where n_{up} is the number of available peers and \bar{F} is the average failure probability of $n - n_{up}$ unavailable peers in the replica group. We pick the m that maximizes $P(x = m)$, *e.g.*, $P(X = m) = \max\{P(x = k) \mid k = 0, 1, 2, \dots, n\}$, and use m as the estimated number of remaining replicas. Since this estimate is the most likely one that occurs in reality, it provides the highest prediction rate (the probability that the estimate correctly matches the reality).

Periodically (*e.g.* every hour), the system uses the estimator to determine whether data recovery is necessary to reach the desired replica target. The system would create new replicas only when the estimated replica number m falls below the target c : If $m < c$, then the replication layer generates $c - m$ new replicas randomly among online peers in the system, otherwise, no action are necessary.

4. PRELIMINARY MEASUREMENTS

A first full prototype of AmazingStore was released for public use on April 3rd, 2009. As of early October 2009, more than 7100 users had registered accounts. The daily peak of simultaneous online users is 780, and is increasing on a daily basis. By default, each user allocates 1GB storage and 2KB/s replication bandwidth (a resource unit defined in the system). In total, the number of files currently stored in the system is above 28600 and continuously rising (not including files such as popular movies that users share each other with file-sharing approach). The total amount of storage currently occupied in the system is roughly 705GB. While the AmazingStore is still growing, we present some initial measurements that show that the system is successfully meeting our expectations.

4.1 Guaranteed System Availability

We first examine the data availability gained on the cloudlet server side. Overall, the system experiences a high data availability of 93.22% (in terms of the fraction of available

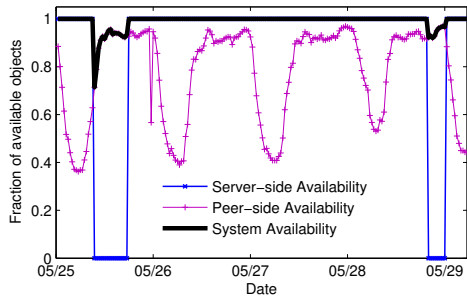


Figure 3: The fraction of available objects over a failure period (May 25-29, 2009), with the details on both server and peer sides. The ticks on the x-axis correspond to midnight on the days that are labeled.

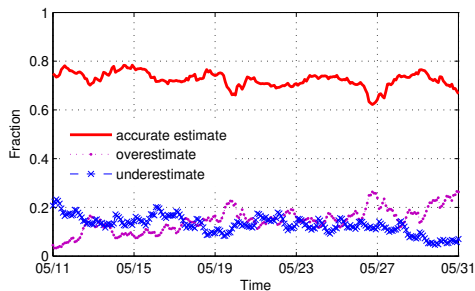


Figure 4: The performance of our replica count estimator over a 20-day period (May 11-30, 2009).

time) in this layer. As expected, the correlated server failures result in unavailability. We illustrate the longest one (an 8-hour power failure from 9:00 am to 5:00 pm on May 25, 2009) in Figure 3. Clearly, we see that one cannot further improve availability by distributing data across more servers due to correlated failures.

On the other hand, our analysis reveals that the average peer is only online 27% of each day. Given two nines availability requirement, the peer layer maintains $c = 6$ replicas (derived based on equation (2)) and provides an average availability of 83.76%. In addition, Figure 3 shows data availability in the peer level exhibits a strong diurnal pattern. As we will show in Section 4.2, this actually improves the offloading efficiency and user experience in that periods of high data availability (9 a.m. to 12 p.m.) is also those of high demand for data access on a daily basis.

We finally see whether the overall data availability (also defined as system availability) meets the target two nines availability. By adding a P2P layer, our analysis shows that the system availability jumps from 93.22% to 99.13% (measured over the whole deployment period). Figure 3 illustrates the availability compensation of the P2P layer. Notice that the system availability is only slightly better than the target, which implies the system incurs very little extra replication cost over long periods. In fact, with a target replica

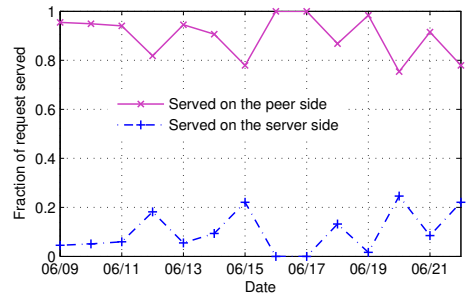


Figure 5: The fraction of request load on both sides over a two-week period (June 19-22, 2009).

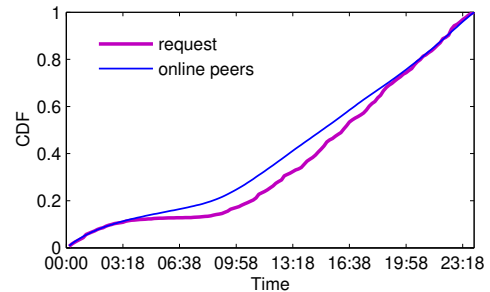


Figure 6: Correlation between peer availability and read requests for data, computed on a daily basis.

count 6, AmazingStore achieves an average of 6.1 replicas per object.

To show how the system achieves a good cost-availability tradeoff on a long-term basis, we also take frequent snapshots during a 3 week period of our deployment (May 11-30, 2009), and look at whether the estimator is accurately predicting the number of alive replicas for each object. At each snapshot, we compute the ratio of data objects for which the estimator was accurate, underestimated, or overestimated the actual replica count. The result in Figure 4 shows that our estimator is highly accurate. As a result, the system is very efficient in repairing failures, and only creates new replicas when replicas are truly lost.

4.2 Low-cost Storage Service

AmazingStore offloads a majority of bandwidth cost by forwarding the requests to the cloudlet peers. Overall, 90.38% of user requests were handled by replicas stored on cloudlet peers, diverting 90.57% traffic load from centralized servers. For the remaining requests, no cloudlet peer replicas were available, and the clients read the data from the centralized servers. Figure 5 shows the fractions of request served on both sides per day over a 2-week period (June 19-22, 2009), in which the requests were highly concentrated (on average 305 requests arrived per day).

A key reason for the high offloading efficiency of the P2P

component is that requests and peer availability have a strong positive correlation. We measure the frequency of data access requests and peer online times on a daily basis, and plot both as CDFs averaged over different time-slices throughout each 24 hour period. As shown in Figure 6, there is a clear correlation between accesses and peer availability, with a correlation coefficient 0.78.

Recall that a centralized cloud still has to maintain multiple replicas (*e.g.*, 3) for high data durability, even though availability cannot be effectively improved due to correlated failures. Thus, the addition of the P2P layer can further reduce the storage overhead of the centralized cloud. Over a half year of deployment, the observed data loss rate in the P2P component of AmazingStore is only 0.07% per month on average, corresponding to a MTTF (Mean Time To Failure) of about 1.05×10^6 hours, which falls within the range of MTTFs claimed by today's highest quality disks [11]. Thus, by taking the P2P layer as a durable disk, the number of replicas maintained in the server layer can be reduced.

5. OPPORTUNITIES & APPLICATIONS

In addition to the above fundamental advantages, the Cloudlets architecture hints to promising directions for further exploration, as well as novel applications deployable on a real platform like AmazingStore. We describe some here.

Network- and Bandwidth-aware Cloudlet Selection. Current replica replacement in AmazingStore is random. By utilizing knowledge of the network through network measurements or an information plane [10], we can optimize applications for different objectives, including resilience to network failures, improved performance through local Cloudlets, and the ability to handle flash crowds and network hotspots.

Socially-aware Cloudlet Management. Integrating social links into the Cloudlets provides another potential dimension for optimization. Prior work has proposed using social links to improve online backup [9]. In our context, using social links can be used to localize data requests within certain communities and improve user privacy.

Lyra: a Distributed Online Social Network. There are a number of disadvantages to centralized social networks such as Facebook and MySpace, including loss of data ownership and lack of data privacy [8]. On the other hand, it is very difficult for "pure" distributed social networks to guarantee data availability, which is critical for friends to browse and share content on social networks. As an alternative solution, we are building a distributed social network, *Lyra*, on the Cloudlet architecture. The *Lyra* system encrypts all private data at the cloud server, and makes private data only accessible after decryption. Key distribution bypasses the cloud server, ensuring that no third party has access to private data. *Lyra* also uses network coordinates to detect locality in data accesses, and biases Cloudlet selection to optimize performance for frequent readers. The Cloudlets infrastructure simplifies data consistency in *Lyra*: all writes are encrypted

and sent to the master replica on the cloudlet server before propagating out to cloudlets.

6. CONCLUSIONS

Empirical evidence has shown us that centralized cloud infrastructure are vulnerable to massive failures. In this paper, we propose an alternative architecture for cloud-based infrastructures that retains the simplicity of management in cloud systems while gaining the failure resiliency available in P2P systems. We advocate that servers in the data center and peers at the network edge complement each other, resulting in highly available and durable systems with lower costs than traditional cloud-based designs. We describe a deployed prototype of AmazingStore, an online storage system adopting the Cloudlets hybrid architecture. While AmazingStore is not yet a mature system, initial results show that its cloudlet peers can improve system resilience by complementing the centralized storage servers. We believe that the properties demonstrated by Cloudlets open up new opportunities for improving storage services and enabling new distributed applications in the cloud infrastructures. The latest version of AmazingStore is available for download at our website at <http://en.amazingstore.org/>.

7. REFERENCES

- [1] BHAGWAN, R., ET AL. Understanding availability. In *Proc. of IPTPS* (2003).
- [2] BLAKE, C., ET AL. High availability, scalable storage, dynamic peer networks: Pick two. In *Proc. of HotOS* (2003).
- [3] CHUN, B.-G., ET AL. Efficient replica maintenance for distributed storage systems. In *Proc. of NSDI* (May 2006).
- [4] FREEDMAN, M. J., ET AL. Democratizing content publication with coral. In *Proc. of NSDI* (2004).
- [5] HUANG, Y., ET AL. Challenges, design and analysis of a large-scale p2p-vod system. In *Proc. of SIGCOMM* (2008).
- [6] KATZ, L. Seattle fire knocks out service to bing travel, other sites. CNet News, July 2009.
- [7] KRAZIT, T. Google networking error caused outage. CNet News, May 2009.
- [8] LAM, M. Building a social networking future without big brother. POMI 2020 Workshop, Stanford, CA, April 2009.
- [9] LI, J., AND DABEK, F. F2F: reliable storage in open networks. In *Proc. of IPTPS* (Santa Barbara, CA, Feb. 2006).
- [10] MADHYASTHA, H. V., ET AL. iplane: An information plane for distributed services. In *Proc. of OSDI* (2006).
- [11] MAYMOUNKOV, P., ET AL. Disk failures in the real world: What does an mttf of 1000000 hours mean to you? In *Proc. of FAST* (2007).
- [12] NEEDLEMAN, R. Minor outage at facebook monday. CNet News, January 2009.
- [13] NEEDLEMAN, R. Rackspace suffers outage. CNet News, June 2009.
- [14] NEWMAN, H. The real cost of storage. Enterprise Storage Forum, April 2007.
- [15] SOBEL, J. Scaling out. <http://www.facebook.com/notes.php?id=9445547199>, August 2008.
- [16] SUN, Y., ET AL. Fs2you: Peer-assisted semi-persistent online storage at a large scale. In *Proc. of INFOCOM* (2009).
- [17] TIAN, J., ET AL. Probabilistic failure detection for efficient distributed storage maintenance. In *Proc. of SRDS* (2008).