Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Optimizing storage performance in public cloud platforms*

Jian-zong WANG[1,2], Peter VARMAN[3], Chang-sheng XIE‡[1,2]

([1]*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*)

([2]*Wuhan National Laboratory for Optoelectronics, Wuhan 430074, China*)

([3]*Department of Electrical and Computer Engineering, Rice University, Houston 77005, USA*)

E-mail: jzwang@smail.hust.edu.cn; pjv@rice.edu; cs_xie@hust.edu.cn

Received Apr. 14, 2011; Revision accepted Sept. 15, 2011; Crosschecked Nov. 4, 2011

Abstract: Cloud computing is an elastic computing model where users can lease computing and storage resources on demand from a remote infrastructure. It is gaining popularity due to its low cost, high reliability, and wide availability. With the emergence of public cloud storage platforms like Amazon, Microsoft, and Google, individual applications and enterprise storage are being deployed on Clouds. However, a serious impediment to its wider deployment is the relative lack of effective data management services. Our experiments, as well as industry reports, have shown that the performance and service-level agreement (SLA) cannot be guaranteed when the data is served over public Clouds. The relatively slow access to persistent data and large variability in cloud storage I/O performance can significantly degrade the performance of data-intensive applications. This paper addresses the issue of I/O performance fluctuation over public cloud platforms and we propose a middleware called CloudMW between the Cloud storage and clients to provide the storage services with better performance and SLA satisfaction. Some technologies, including data virtualization, data chunking, caching, and replication, are integrated into CloudMW to achieve a more stable and predictable performance, and permit flexible sharing of storage among the virtual machines (VMs). Experimental results based on Amazon Web Services (AWS) show that CloudMW is able to improve the stability and help provide better SLAs and data sharing for cloud storage.

Key words: Cloud storage, Performance fluctuation, Middleware, Service-level agreement
doi:10.1631/jzus.C1100097 **Document code:** A **CLC number:** TP393

1 Introduction

Cloud computing is an emerging Internet-based computing paradigm whereby clients can lease shared computational resources, storage, and software on a pay-as-you-go (PAYG) basis (Armbrust *et al.*, 2009). Users are spared the complexities of ownership of the computing infrastructure, and avoid the costs of buying and maintaining their own equipment, providing floor space, infrastructure management, upgrades, electricity, and cooling. In traditional private data centers, the need to handle critical peak loads usually leads to significant over provisioning of server capacity, resulting in low overall utilization. By contrast, the elastic nature of cloud services that allows resources to be dynamically added and removed based on the actual demand, makes it an economically attractive model. Many companies such as Amazon, Google, and Microsoft have recently launched commercial cloud services: clients rent cloud resources for both computing cycles and long-term data storage. The economies of scale of consolidated infrastructure, the use of commodity resources, improvements in networked connectivity, and the growth in the capabilities of personal mobile devices are converging to make cloud services a viable business model, which could shape future

‡ Corresponding author

* Project supported by the National Basic Research Program (973) of China (No. 2011CB302303), the National High-Tech R&D Program (863) of China (No. 2009AA01A402), the National Natural Science Foundation of China (No. 60933002), the Chenguang Plan of Wuhan, China (No. 201050231073), the Innovation Plan of WNLO, and the National Science Foundation of USA (Nos. CNS-0917157, CNS-0615376, and CNS-0541369)

computing systems. Spending on cloud services by companies is projected to reach $42.1 billion in 2012, growing by a factor of 27% per year (International Data Corporation, 2010).

1.1 Motivation

Resource sharing in a cloud is facilitated by advances in server and virtualization technologies. High performance servers provide the computing power to simultaneously handle the workloads of multiple client applications. Virtualization allows the different applications to be flexibly encapsulated within virtual machines (VMs) that multiplex the CPU cycles and memory of the physical host server. VMs are also allocated storage volumes, which may reside on the disks attached to the host or in a disk array shared by several physical hosts. This allocation is typically transient, in that the allocated space is released along with the associated VMs; hence, data that needs to persist across VM instantiations must be saved on more permanent storage before VM exit and read back in from the permanent store on VM start up.

Efficiently managing cloud resources and maintaining service level agreements (SLAs) for cloud services is a tremendous challenge. Performance virtualization techniques have been successfully employed to provide effective multiplexing of CPU and memory resources (as in VMware's ESX servers) with predictable performance (VMware ESX, 2010). However, providing performance guarantees for shared storage and I/O resources (Gulati *et al.*, 2007; 2010) is a difficult problem that remains an area of active research. The problem is further exacerbated in cloud environments due to their scale, heterogeneity, and variable Internet network latencies. Consequently, data access times tend to be high and exhibit large variability, which is unacceptable in a number of applications.

One of the most representative cloud computing platforms is Amazon Web Services (AWS) (Amazon AWS, 2010). AWS provides leased computational power (elastic cloud computing or EC2) in the form of VMs with different processor/memory capacities, optionally pre-configured with different operating systems and application software. In addition, AWS provides a number of options for cloud storage, specifically: Amazon simple storage service (S3), Amazon elastic block storage (EBS), and EC2 local disks (ELD), which can be used to store data. These three storage mediums have different characteristics (Table 1).

Amazon S3 is the durable and persistent cloud storage platform for AWS. S3 provides reliable long-term data storage with convenient and ubiquitous access using a simple Web interface. A major advantage of S3 is its support for global data sharing that allows the same data volumes to be simultaneously accessed from different locations or different VMs. However, several experiments have shown that the performance of the S3 storage service is not as good as those of other storage alternatives in terms of both throughput and predictability.

We considered a simple file transfer operation to validate this hypothesis. We measured the transfer bandwidth by downloading a 1 GB file from Amazon S3 to Amazon EC2. Table 2 shows the download time and corresponding transfer rate (TR) for 20 separate download experiments. Fig. 1 shows the same data as a cumulative distribution function of download time (a standard data fitting program was used to smoothen the measured data). It is observed that the throughput fluctuates between 10 and 20 MB/s, sometimes falling even lower than 10 MB/s. The network bandwidth limit of a small machine instance of EC2 (see Section 2) that acts as the the destination node is about 25 MB/s. This motivates the development of architectures and mechanisms to improve the

Table 1 Characteristics of cloud storage media on Amazon Web Services

Attribute	Simple storage service	Elastic block storage	EC2 local disks
Storage capacity	Unlimited	1 GB to 1 TB	160 to 1690 GB
Durability	99.999999999%	Not guaranteed	Not guaranteed
Availability	99.99%	Not guaranteed	Not guaranteed
Data life cycle	Forever	Forever	Till EC2 terminates
Data sharing support	Yes	No	No
Cost	$0.150/GB	$0.10/allocated GB	$0.085/hour
Performance	Fluctuates	Small variation	Stable

performance and stability of cloud data access over public Clouds.

Table 2 Variance in transfer time for a 1 GB file

Test No.	Time (s)/ TR (MB/s)	Test No.	Time (s)/ TR (MB/s)
1	83/12.3	11	84/12.2
2	73/14.0	12	85/12.0
3	98/10.5	13	49/20.1
4	77/13.3	14	55/18.6
5	104/9.85	15	70/14.6
6	80/12.8	16	84/12.2
7	69/14.8	17	61/16.8
8	77/13.3	18	90/11.4
9	97/10.5	19	91/11.3
10	95/10.8	20	102/10.0

TR: transfer rate

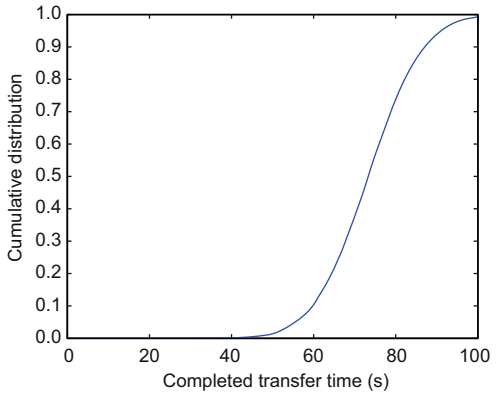

Fig. 1 Cumulative distribution function of transfer time

1.2 Basic idea and contributions

The motivation of this paper is to fully utilize the storage throughput of cloud storage and reduce the performance variability in a cloud computing platform. The basic idea is to provide a middleware layer, called CloudMW, to engineer a hybrid cluster storage architecture that combines durable storage (S3 in AWS) for long-term data, persistent EC2-attachable storage (EBS) for active data sets, and VM-instance local disk storage as local caches. Our overall goal is to investigate the design of an effective hybrid cloud storage architecture that balances cost, performance, predictability, and reliability. The main contributions of this work are listed below:

1. We identify and address problems in performance in cloud storage, and propose a hybrid cluster storage architecture to optimize the performance over public Clouds.

2. Provide a middleware layer that can provide better performance through the use of caching, data chunking, and parallel transfer to increase throughput, and exploit data replication to achieve lower variance in data transfer times.

3. Study various topological structures for data sharing on public clouds, and investigate their performance using CloudMW for data-intensive applications and benchmarks.

4. Show how potential schemas for performance enhancement (for instance, the cooperative competition strategy) can be integrated into CloudMW to guarantee the SLAs of cloud storage services.

2 Background

2.1 Cloud storage

Cloud storage provides a reliable and transparent storage back end in a data center. Cloud storage is usually distinguished from specialized data access services like cloud backup in providing a general access interface and flexibly expanding the amount of available storage space. It supports specific application program interfaces (API) for direct access by other services: for instance, Amazon S3 can afford peer-to-peer (P2P) services to act as a BitTorrent seed sender at the cloud end (Amazon AWS, 2010).

2.2 Amazon elastic cloud computing (EC2)

Amazon elastic compute cloud (Amazon EC2, 2010) is a Web service that provides access to virtual-machine-based computing cycles within the cloud. A client may choose from a number of pre-defined machine configurations depending on their application. For example, EC2 defines small, large, and extra large machine instances based on the memory size (1.7–15 GB), number of virtual cores (1 to 4), each with the capacity of a 1.0–1.2 GHz 2007 Xeon or Opteron processor, local storage (160 to 1690 GB), and a 32- or 64-bit platform. Other specialized configurations for memory- or computation-intensive applications are also available. Machines can be equipped with a choice of operating systems and software to ease the deployment of client applications. A variety of pricing options are provided ranging from an hourly rate based on the configuration, to a reserved-instance rate based on an

up-front cost and a discounted hourly charge. Amazon provides a simple Web service interface to configure capacity; dynamic scaling is eased by the relatively short time (minutes) required to obtain and boot new server instances.

2.3 Amazon simple storage service (S3)

Amazon S3 (Amazon S3, 2010) is a cloud storage platform used for persistent data storage and convenient ubiquitous access using a simple Web interface. Costs are kept low by using commodity hardware, and data is replicated to increase availability and provide reliability against failures. S3 provides basic functionality to read, write, and delete objects ranging from 1 byte to 5 GB. Pricing monthly on a per GB basis depends on the geographical location of the storage; data transfers across Amazon S3 regions may also incur per-request and transfer-volume based costs. SLAs govern the reliability and accessibility of the data, but do not provide performance QoS guarantees. S3 and EC2 together provide a comprehensive cloud storage and cloud computing platform.

2.4 Amazon elastic block store (EBS)

Amazon EBS provides block level storage volumes for use with Amazon EC2 instances. Amazon EBS volumes are persistent and, unlike local EC2 storage, have a lifetime independent of any EC2 VM instance. An EBS volume can be attached to a running Amazon EC2 instance and exposed as a device within the instance (Amazon EBS, 2010). However, EBS has some limitations. First, it lacks reliability guarantees since data is not automatically replicated. In a typical usage scenario, volumes created on EBS can be backed up on S3 using the automatic incremental checkpointing facility. In addition, the EBS interface is closer to direct-attached storage (DAS) than to a network-attached storage (NAS) or storage area network (SAN) architecture. An EBS volume can be attached to only a single VM instance, which alone can read and write the data on the volume. Hence, it is not possible to automatically share data on the volume among different applications running on different EC2 VMs. CloudMW presented in this paper is a middleware layer that provides the functionality to allow EBS volumes to be shared among many client VMs.

2.5 I/O virtualization

Cloud computing makes heavy use of virtualization technology to share resources without conflict. I/O contention is a significant bottleneck among VMs sharing a physical host and can impact the performance of data-intensive computations significantly. Currently, XEN is the most popular open source software to deploy on public clouds, for instance, AWS. The I/O virtualization in XEN where all I/Os go through the driver domain limits the performance of individual VMs. Applications running in the cloud computing framework suffer from I/O virtualization bottlenecks owing to the contention among the virtual machines.

3 System architecture

In this section, we present an overview of our architecture CloudMW. In our model, cloud storage provides data services for the virtual machines leased by the customers. We propose a hybrid storage architecture using S3, EBS, and EC2 local storage, and middleware to support efficient data transfers between S3 and EBS, and data sharing of the EBS volumes among multiple EC2 VM clients. Therefore, we divide the framework into two layers: cloud management and storage management middleware, which are responsible for data transfer and data sharing, respectively (Fig. 2). These two middleware layers address different functions: the cloud management layer deals with the EBS to S3 interface; the storage management provides the interface between VMs running on EC2 and the shared data on EBS volumes.

The cloud management layer of CloudMW is responsible for storing data and providing data services using the service-provider defined API. It can also handle data encryption, data decryption, data backup, data restoring, data deduplication, and data consistency for the data transferred between S3 and EBS. As mentioned earlier, the performance of S3 is not sufficient to maintain the throughput of data intensive applications. Hence, working data sets will be stored on EBS volumes and backed up to S3 as desired for reliability (replicas maintained by either S3 or CloudMW) and archival. The storage management layer will provide data services for client virtual machines. Our proposed framework will

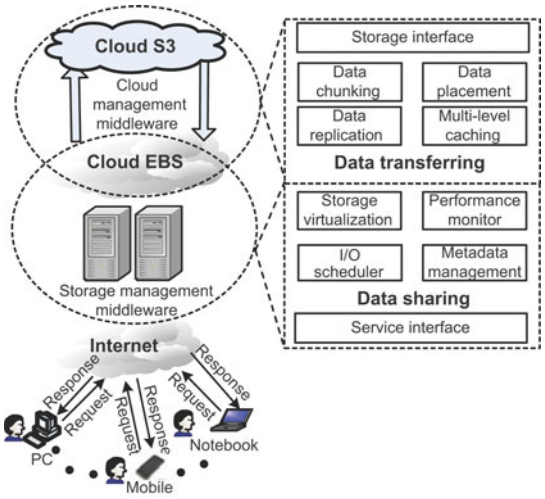

Fig. 2 Architecture of the proposed CloudMW system

provide for sharing and enhanced performance using a variety of mechanisms: data chunking, data placement, data replication, storage virtualization, multi-level caching, performance monitoring, I/O scheduler, and metadata management. We provide some details of these components below.

Data chunking: In a conventional cloud environment the storage node is mounted on the local file system of the EC2 VM and read directly. However, our experiments show that S3 is unable to saturate even a small EC2 machine instance, achieving a throughput between 10 and 20 MB/s for file transfer, compared to the EC2's peak capacity of around 25–30 MB/s.

Data replication: Replication is a cornerstone of reliable distributed storage and can also be leveraged for enhanced performance. The data duplicator can partition the file into replicated chunks that are striped across S3/EBS nodes. Different replicas may also be stored in different providers for enhanced reliability. This schema can achieve the redundancy necessary to tolerate individual failure, and alleviate performance bottlenecks thereby decreasing the fluctuation in performance. The unit must also ensure the consistency of multiple write copies.

Data placement: This component is acting as the conductor to implement data locality and maximize storage throughput. When activated by the performance monitor, the migration thread will assign the data to the best locations based on the appropriate chunking and replication policy in effect. The unit decides whether to partition a file, and if so, the number and location of the chunks. It also decides how many replicas to make and their locations. Different replicas may also be stored in different providers for enhanced reliability.

Storage virtualization: To access and manage the cloud storage, we integrate the Filesystem in Userspace (FUSE) (FUSE, 2010) into CloudMW. Using this interface, the cloud storage space can be virtualized and viewed as a single storage device. Clients execute all read and write operations through FUSE.

Performance monitor: The monitor tracks the real-time performance of VMs and S3/EBS storage and detects performance trouble spots. Once the bottleneck is identified, it will inform the module of data placement to place new data and even migrate existing data to other locations with better performance. The cost overhead of the monitor deployment is small, and can result in significant performance benefits.

Multi-level caching: The tiered architecture consists of several types of storage. S3 provides the back-end long-term storage. On-line access to the data is from EBS volumes on which the hot portions of the data from S3 are stored. Client VMs have direct-attached storage which provides the fastest access to data private to that VM. Caching technology must be adopted in the framework where S3 acts the secondary storage medium and EBS provides on-line storage services. Meanwhile, the leased EC2s inside the CloudMW can also be provisioned into the services to utilize their better access performance.

I/O scheduler: The I/O scheduler handles issues such as locality of accesses across VMs, high variability in I/O sizes, different request priorities based on the applications running in the VMs, and bursty workloads. The amount of I/O throughput available to any particular host can fluctuate widely based on the behavior of other hosts accessing the shared device. The system must be able to fulfill the scheduling of I/O requests to balance the system workload.

Metadata management: Data stored in S3 is organized over a two-level namespace. In this proposed framework, we will use SOAP/REST protocols to communicate and manage the metadata information of the files storing in the cloud. The metadata manager uses novel metadata management mechanisms to generate small-sized indirection maps to translate file system level block addresses to the

current locations of those blocks on the cloud storage and EC2's local storage.

4 System strategies

In this section we describe the data transfer and data sharing strategies used by CloudMW.

4.1 Data transfer strategies

We use the file transfer process as an example to show the basic data transfer strategies that are supported in CloudMW.

Data chunking and replication: In a conventional cloud environment, the storage node is mounted on the local file system of the VM and read directly as shown in the first diagram in Fig. 3. As our initial experimental result in Fig. 1 showed for the S3/ESB cloud storage system, S3 is unable to saturate even a small machine instance of VM2, achieving a throughput between 10 and 20 MB/s for the file transfer, compared to the VM's peak capacity of 25 MB/s. To improve the performance, middleware will automatically partition a file into n chunks that are striped across independent storage nodes. A file transfer operation is intercepted by the middleware, which translates the request to concurrent data transfers for the n chunks comprising the file. In our experiments for a 1 GB file transfer, splitting the data into three independent chunks that are accessed concurrently saturated the VM and maximized the peak aggregate data transfer bandwidth. This parallel access is a well-established strategy underpinning both array-based RAID storage and distributed data systems like OceanStore (Kubiatowicz *et al.*, 2000). However, our experience indicates that in a cloud computing environment the strategy is insufficient to obtain consistent performance. In fact, if any one of the storage nodes is sluggish, the entire data transfer is delayed and the aggregate bandwidth is reduced.

To avoid performance fluctuation, data will be replicated over a number of nodes. Each replica is read concurrently by independent helper nodes (VMs), and the winner transfers the data to the destination over cluster network links whose bandwidths in the EC2 environment have been shown to be stable (Wang and Ng, 2010). The transfer time from cloud storage to the computing cloud is governed by the transfer time of the fastest of the replicas and has a much smaller variance than the original single source transfer. A simple analytical argument to show the potential improvement in the stability of the data access time is discussed below.

Assume that the transfer time of a file from a single storage node is governed by the cumulative distribution function (CDF) F; i.e., the probability that the transfer time T is less than or equal to t is given by $F_{\mathrm{T}}(t)$. First, let the file be duplicated over k independent storage nodes with the same transfer time probability distribution. Let T_i denote the random variable representing the transfer time of the ith storage node. The transfer time achieved by the replication strategy is $D = \min_{1\le i\le k} T_i$. Now, the probability that $D \ge t$ is the probability that all k transfers require at least time t (i.e., $\{T_1 \ge t\}, \{T_2 \ge t\}, \ldots, \{T_k \ge t\}$), which is $(1 - F_{\mathrm{T}}(t))^k$. As can be seen, this goes down exponentially with the number of replicas. For example, assuming a uniform probability distribution for the transfer time, for $k = 1$ the probability of exceeding 150% of the mean is $1 - 0.75 = 0.25$ or 25%, for $k = 2$ the probability decreases to 6.25%, and for $k = 3$ the probability reduces further to 1.5%. As can be seen, the variability in the transfer time is significantly reduced. The entire data transfer process is described in Fig. 3. The file is partitioned into n chunks and each chunk is replicated k times (in our experiments we used

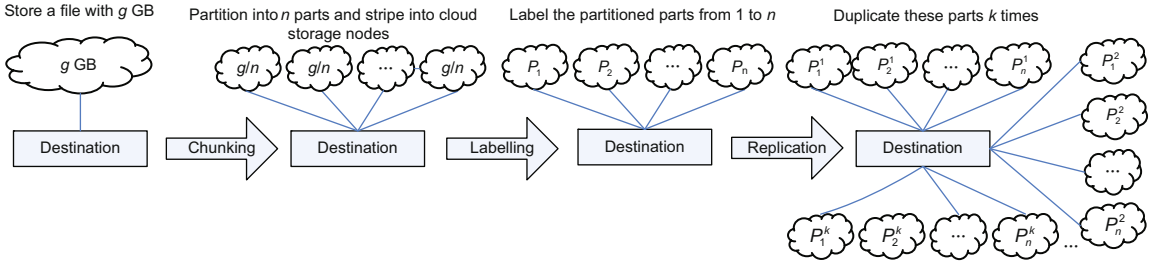

Fig. 3 Chunking and replication

$n = k = 3$).

Multi-tier architecture: A multi-tier architecture for coordinating the data transfer is shown in Fig. 4. In applications such as file downloading, the data can be accessed through the middleware layer acting as a relay. The great advantage of adopting this kind of design is to use the stable performance between EC2 nodes inside CloudMW to mitigate the unstable performance of S3 to EC2 using the replication strategy described above.

The cooperative competition protocol: The cooperative competition protocol is used to affect the data transfer using the multi-tier architecture. In Fig. 5, k (equal to 3 here) helper VMs E_1, E_2 through E_k are used to speed up the transfer. E_i is responsible for requesting the ith replica of any chunk. This solution works in rounds: in each round a chunk is transferred simultaneously to each node E_j from the jth storage node storing the chunk. When the transfer to any E_i is complete, the completed node notifies each of the other E_j's, which then cancel their transfer if it is still in progress. A static rule is used to break ties unambiguously. Each E_j then requests the next chunk. When all chunks have been received, they are transferred to the destination. Several optimizations and variants of this basic scheme can be considered. The transfer of a chunk to the destination from the winning E_i in the previous round can be overlapped with the transmission of the current chunk from S3. In the above scheme it is possible that some E_i is the winner in many (or all) of the rounds. This will lead to load imbalance in the second phase while transferring the data from the intermediate EC2 node E_i to the destination. The significance of the problem depends on the saturation throughput of the destination node relative to E_i. If both nodes have the same bandwidth, then the destination node is the bottleneck and there is no effect from reducing the concurrency. If the destination node has a larger bandwidth (a large or extra-large VM, for instance), then some mechanism to spread the chunks among the E_i will result in a better

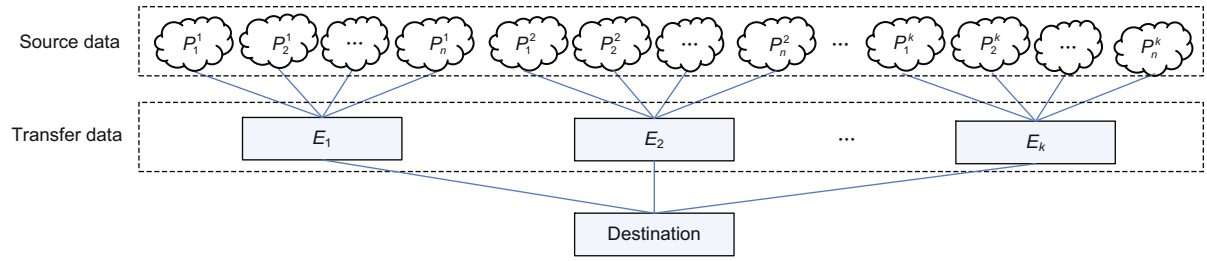

Fig. 4 The multi-tier strategy

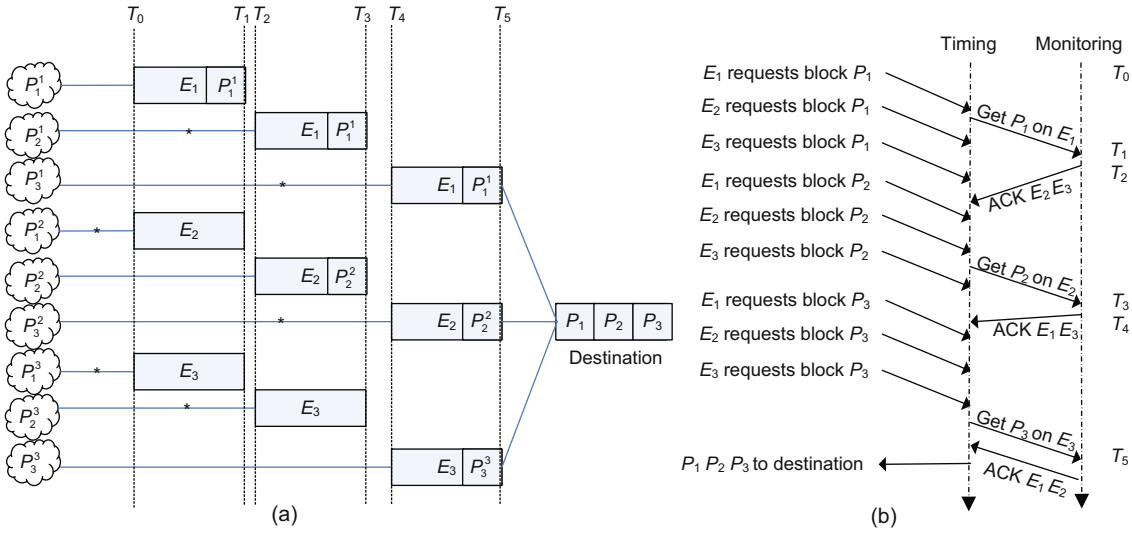

Fig. 5 An example showing $k = 3$ EC2s transferring $n = 3$ chunks of data (a) where P_i^j is the jth replica of the ith chunk, and the protocol expression of cooperative competition processing (b)

performance. Simply changing the protocol so that, in different rounds each E_i fetches from a different storage node, avoids systematic bias arising from the location of a storage node, thereby spreading the chunks among the E_i. Finally, one can consider using kn nodes $E_{i,j}$ to simultaneously transfer k copies of each of n chunks to the intermediate nodes. As before, the winner of each chunk transfers the chunk to the destination. In this situation each chunk will always be transferred to the destination by a different node. The idea is to tradeoff between cloud resources which are cheap and balance out the negative effects of performance fluctuations in the cloud.

Two metrics, completion time γ and throughput β, are defined to reflect the storage performance (Table 3). The completion time consists of two parts: the time transferring data from the cloud storage provider to EC2 intermediate nodes and then to the destination. Simple formulae are derived below to bound these metrics. In the formulae, it is assumed that the receiving node has not saturated its capacity; otherwise, the capacity of the receiver determines the performance. The file of g MB is divided into n parts, so a deterministic upper bound for the time of phase 1 ideally equals $\sum_{i=1}^{n}(g/(nH_{\mathrm{s}i}))$. By using the protocol, it is expected that the actual time match this bound probabilistically in the vast majority of cases. The time for the second phase is upper bounded by $\sum_{i=1}^{n}(g/(nH_{\mathrm{e}i}))$. The overall time γ is either the sum or the maximum of the two bounds depending on whether the faces are overlapped or not. Finally, $\beta = g/\gamma$.

Table 3 Nomenclature

Parameter	Unit	Description
g	GB	The size of files
n	1	Number of segments or S3 nodes
k	1	Number of replicas
m	1	Number of E_i in the transfer layer
m_n^k	1	Label of the kth replica of E_n
H_{s}	MB/s	S3 to EC2 throughput (unstable)
$H_{\mathrm{s}i}$	MB/s	S3 to E_i throughput
H_{e}	MB/s	Inter EC2 throughput (25 MB/s)
$H_{\mathrm{e}i}$	MB/s	E_i to EC2 throughput
γ	s	Total transfer time
β	MB/s	Throughput for file transfer
P_{s}	Dollar	Pricing of S3 (Amazon S3, 2010)
$P_{\mathrm{e}i}$	Dollar	Pricing of the ith EC2 (Amazon EC2, 2010)
$T_{\mathrm{e}i}$	s	Service time of E_i
δ	Dollar	Total cost

4.2 Cost discussion

Pay-as-you-go (PAYG) is an important feature of cloud computing. AWS allows customers to add or remove resources as needed based on the demands of the applications. AWS also offers a reserved instance cost model consisting of a fixed cost plus a lower per-hour usage rate. Currently, many companies employ Amazon EC2 and S3 to power a host of on-demand application servers in order to save money. The cost using cloud computing can in some cases be significantly lower than that using the traditional approach where the customers incur both capital expenditure to buy hardware and continual operational costs for maintenance, energy, space, etc. AWS increases the shareability and flexibility in data use at a nominal cost. The additional cost is that of the server VMs in CloudMW. However, the number of such VMs needed scales at a rate no faster than that of the number of client VMs, in keeping with the PAYG cost structure. A rigorous economic model/analysis is beyond the scope of this paper. Determining the best cost (δ)/performance (β) point of a solution as a function of the cost and transfer parameters and choice of protocol is an avenue for future research.

4.3 Data sharing strategies

4.3.1 Simple sharing strategy

Data sharing is fundamental to many data intensive applications, ranging from transaction processing to data analysis, which operate on common data sets. In the AWS environment, an EBS data volume can be accessed only by the single VM on which it is mounted, making it unsuitable for direct use in these applications. On the other hand, shareable S3 storage has poor performance characteristics, which makes it more unsuitable for on-line storage.

The data sharing over Amazon AWS is done by static replication of snapshots (Fig. 6). In the example, an EBS volume (source storage) is mounted on the single EC2 VM instance shown in the left of the figure. Three other EC2 VMs (in the right of the figure) wish to access the data set on the EBS volume, but are prohibited from directly accessing the volume due to limitations in the cloud infrastructure. Existing solutions handle this situation by making multiple copies of the source EBS volume, one for each VM that wants access. This is done by using

the snapshot feature of AWS to make a copy of the EBS volume on S3 and then to copy the volumes on each of the desired EBS volumes.

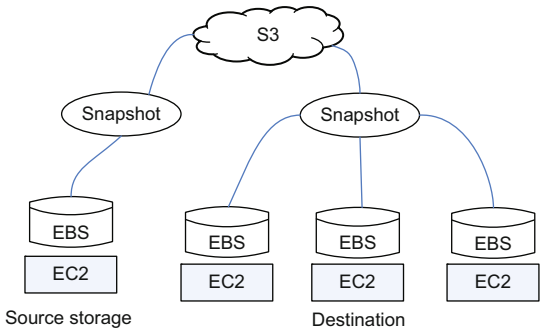

Fig. 6 Sharing an elastic block storage (EBS) volume by replication

There are several limitations with this approach. First, data transfer to and from S3 is slow, and while incremental backups to S3 can reduce the traffic between EBS and S3, the traffic to a freshly instantiated EBS volume from S3 is time consuming. Making multiple copies further slows the transfer and costs are increased due to redundant volumes. Furthermore, if the original data set is spread over several EBS volumes (either due to the size of the data set or to increased performance), the problem is proportionally magnified, since each volume needs to be replicated. Finally, if data is not read-only, then the isolation between the copies precludes updates from being seen across VMs and introduces the possibility of fatal data consistency problems.

It is worth thinking about new ways to avoid these obstacles. The CloudMW proposed here is a step in this direction to allow multiple VMs to share multiple EBS volumes in a flexible and efficient manner.

4.3.2 Middleware enabled sharing strategy

Fig. 7a shows the middleware CloudMW interposed between the EBS volume and a client. CloudMW instantiates an EC2 virtual machine that is attached to the EBS volume, which acts as a router or switch for the clients. FUSE (FUSE, 2010) is used to connect the EC2 switch to the client VM. In the data sharing strategy, our experiments of transferring a 1 GB file from the EBS volume to the local attached disk of the client (ELD) achieve a throughput of roughly 22 MB/s. Increasing the number of EBS volumes that feed the client does not change this

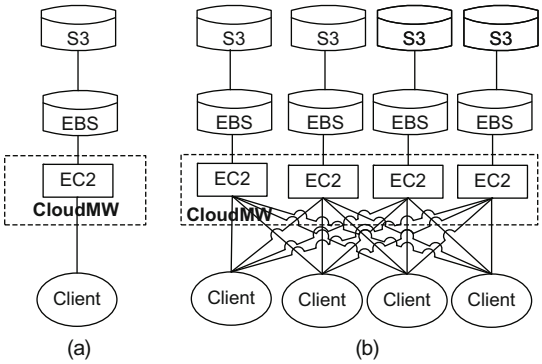

Fig. 7 Middleware enabled data sharing: (a) single S3 and EBS shared by a client; (b) multiple S3 and EBS shared by multiple clients after each client VM adds a new EC2 switch to the middleware

throughput appreciably. We attached three EBS volumes to the EC2 in CloudMW in the above example. The client requested 1/3 GB transfer from each of the three EBS volumes. The total time was changed only a little and the net throughput for the transfer was around 26 MB/s. This suggests that the EC2 to EC2 bandwidth between the VM in CloudMW and the client VM limits the transfer rate; hence, striping or chunking a file across many EBS volumes by itself is insufficient to speed up the bandwidth to a single client. We also experimented with using a single EBS volume to transfer data to several clients simultaneously using CloudMW. We found that the aggregate bandwidth out of the EC2 switch in CloudMW was roughly the same (approximately 26 MB/s), and was divided roughly equally among the client VMs attached to the switch.

These performance results led to the proposed CloudMW architecture shown in Fig. 7. The number of EC2 switches in CloudMW is equal to the number of clients to prevent the EC2 bandwidth from becoming a bottleneck. In our initial design, there is a full connection between the clients and the switches. For a large number of clients, a hierarchical switching topology may be more appropriate, and we are investigating this currently.

Experimental results for different data transfer and sharing operations are described in Section 5.

5 Empirical evaluation

In this section, we evaluate our framework in the AWS cloud using S3, EBS, and EC2 services. We register S3 storage space and lease several small EC2 instances to act as destination and helper VMs.

CloudMW is implemented in our framework using leased EC2 VM small instances that are attached to the EBS volumes. Note that an EBS volume may be attached to at most one EC2 instance at any time.

In other words, our experimental testbed consists of multiple S3 cloud storage nodes, multiple EBS volumes, multiple EC2 instances, and multiple destinations that are also small EC2 instances. One can also lease other kinds of EC2 to satisfy specific application requirements; for example, the throughput can be doubled by the deployment of the large EC2 instances.

5.1 Basic EC2 and S3 I/O performance

We have measured several I/O performance metrics on Amazon EC2 instances, including the throughput between S3, EBS, and ELD for data transfer operations. Our experiments are focused mainly on small EC2 instances, which are the default instance type of Amazon EC2. We configure one EBS volume with 2 GB on an EC2 VM, which can also access S3 using a FUSE mount point. The test measures the data transfer rate between different mediums. We also test the network I/O performance of interconnected EC2s by transferring between the ELDs of two EC2s. Fig. 8 introduces the experimental configuration, whose results are summarized in Table 4. Each row summarizes the range of throughputs measured for reading or writing data between two storage devices. Row 1, S3 to ELD, means the transfer of data between an S3 volume and an EC2 VM's local disk (ELD); 'read' means 'download data from S3 to ELD', while 'write' means 'upload data from ELD to S3'. The setup uses FUSE to mount S3 to EC2's local disk, after which a direct copy between the devices can be effected. Row 2, S3 to EBS, means the transfer of the data between the S3 volume and a direct-attached EBS volume on the EC2 VM. It behaves the same way when accessing the ELD on the VM. In both cases, the read throughput is low and varies between 10 and 22 MB/s, while the write throughput is lower at 5–9 MB/s. As noted earlier, this makes direct on-line access of S3 unsuitable. For row 3 of the table, we mount EBS on EC2 VM and transfer data from EBS to the EC2 VM local disk. This has a much higher bandwidth of 50–56 MB/s for reads and 40–45 MB/s for writes. Row 4, ELD to ELD, means the measure of the data transfer between the local disks of two different EC2 VMs. This actually measures the inter-EC2 network bandwidth, and the transfer rate is quite stable around 24 MB/s. The command for remote copying (SCP) is used to affect the data transfer. Finally, internal ELD measures the performance of local disk copying operations.

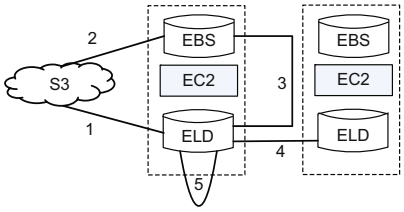

Fig. 8 Basic performance testbed configuration

Table 4 Basic storage I/O performance results

Row number	Test objective	Throughput (MB/s)	
		Read	Write
1	S3 to ELD	10–22	5–9
2	S3 to EBS	10–22	5–9
3	EBS to ELD	50–56	40–45
4	ELD to ELD	23–25	12–16
5	Internal ELD	204	98

Table 4 shows that the performance of data reads and writes between EC2s is quite stable. In contrast, the performance between S3 and EC2 (whether attached EBS or local disks) is lower and highly variable. This validates the assumption and the motivation for using the multi-tiered architecture that migrates the data from S3 to use the better throughput of EBS or local disks. CloudWM addresses the problem of variable performance of S3, which can reduce the time for copying to EBS. In our work, we propose the hybrid architecture to support efficient data transferring between S3 and EBS and data sharing of the EBS volumes among multiple EC2 VM clients.

5.2 Chunking and replication testing

We first experiment to find out how many S3 nodes should serve a single EC2 node to saturate its bandwidth. Fig. 9 shows the results of partitioning the data into one to six chunks. The destination node reads each chunk concurrently. When the file is not partitioned the access time varies within 70–85 s (variable with different experimental runs). Partitioning reduces the transfer time, but the gains after three partitions are relatively small.

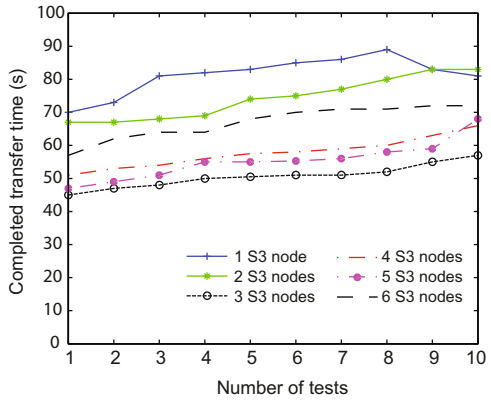

Fig. 9 Access time of partitioning the data into one to six chunks

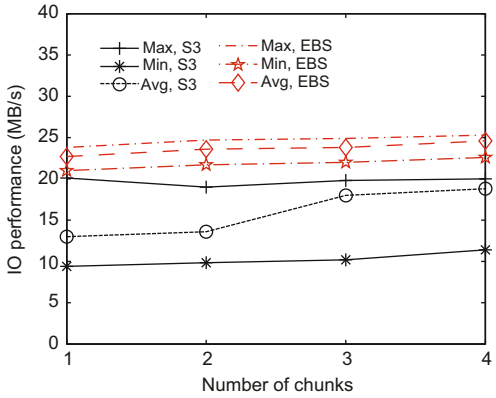

Fig. 11 I/O performance of chunking over S3 and EBS

Our experiments are configured to use $n = k = 3$. The results of the framework are illustrated in Fig. 10. By adopting our proposed schemas, the distance between maximum and minimum throughputs decreases. Thus, CloudMW can obtain more stable and better performance compared with the original architecture, as we have expected.

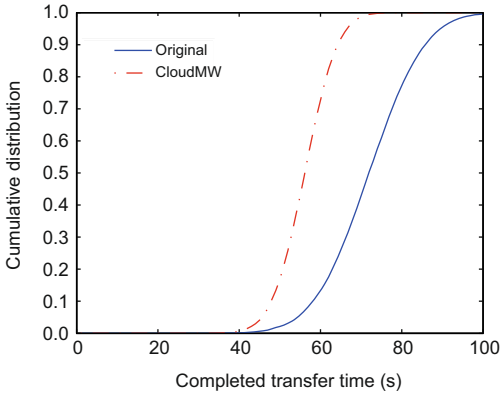

Fig. 10 Chunking evaluation results

In addition, we evaluate the benefits of using CloudMW to partition (chunk) the data and store it on different EBS volumes. The test is based on the architecture as shown in Fig. 7a.

To carry out the chunking schema, the throughput improves according to the increase in the number of chunks, which makes the performance facilitate SLA guarantees (Fig. 11). The empirical results are shown in Table 5.

5.3 Benchmark evaluation

In this subsection, we evaluate the sharing middleware using benchmark programs. We use Postmark, a widely used file system benchmark tool written by NetApp, INC (Postmark, 1997), to validate our framework. The size of requests in Postmark for reading and writing can be configured to different ranges: we set three file size ranges: 0.5 KB to 100 KB, 100 KB to 200 KB, and 200 KB to 1 MB. We configure the number of transactions to 500, and the number of generated files is also set to 500. First, we use one client to send the Postmark requests through CloudMW to access either EBS or S3. We then increase the number of clients, from one to three based on the architecture in Fig. 7a. The comparison of the performance between S3 and EBS is shown in Table 6. The results show clearly that EBS is vastly better in both read and write performance. In addition, since the aggregate throughput is not increasing significantly as we increase the number of clients, the system bottleneck is in the storage subsystem. In particular, the single EC2 VM in CloudMW in this configuration is the bottleneck. To test the scalability, we evaluate the workload for each of the remaining configurations in Fig. 7, except that a single client VM is used to run Postmark and access the data in all cases. As shown in Fig. 12, read and write I/O performance increases with the increase of the number of EC2 VMs in CloudMW. The result is encouraging in that the bottleneck inside CloudMW can be removed. We conjecture that the write performance is superior due to write buffering.

5.4 Throughput and scalability evaluation of CloudMW

To evaluate the scalability of CloudMW, we test the performance of transfer of 1 GB of data on a shared storage pool of EBS volumes extending to

Table 5 Chunking test results

Number of chunks	Chunk size	Average time (s)	Average throughput (MB/s)
1	(1, 1) GB	(75, 45)	(13.6, 22.7)
2	(0.5, 0.5) GB/chunk	(78, 43)	(13, 23.6)
3	(0.33, 0.33) GB/chunk	(54.5, 43)	(18.8, 23.8)
4	(0.33, 0.25) GB/chunk	(57, 41)	(18, 24.6)

In (x, y), x corresponds to storage S3, and y to EBS

Table 6 Comparison of read/write I/O performance between S3 and EBS

Number of clients	Throughput* (KB/s)		
	0.5–100 KB	100–200 KB	300 KB–1 MB
1	(47/144, 1730/5222)	(117/376, 2765/8499)	(300/800, 3820/11 468)
2	(30/114, 1520/4633)	(85.5/320, 1843/5939)	(301/1158.5, 2498/7425)
3	(27/124, 1024/3140)	(74/359, 1372/4420)	(262/1087, 1584/4676)
4	(16/115, 721/2204)	(68/292, 884/2852)	(213/991, 911/2700)

* Read/Write. In (x, y), x corresponds to storage S3, and y to EBS

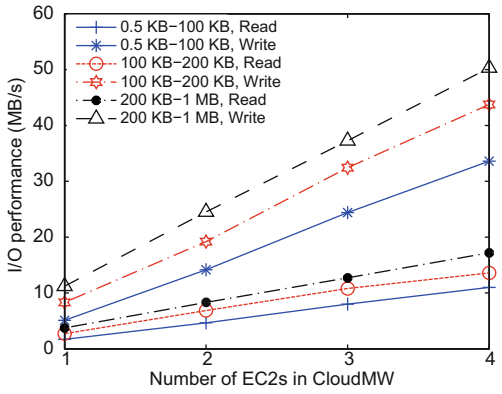

Fig. 12 The I/O performance of Postmark

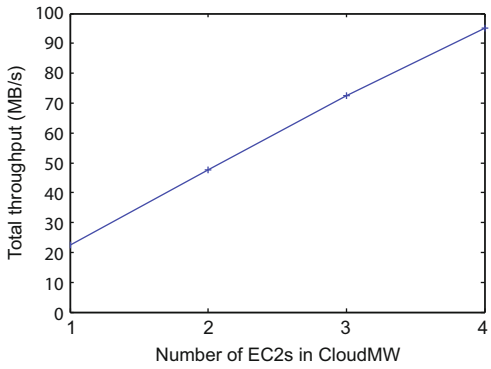

Fig. 13 Throughput and scalability evaluation of CloudMW

a set of EC2s. A 1 GB data set is equally chunked among EBS volumes (between 1 and 4) and each EBS volume is connected to an EC2 VM in CloudMW. Each EC2 in Fig. 7b reads the entire 1 GB of data from all the EBS volumes. As shown in Fig. 13, as the number of EC2s is increased, the aggregate throughput of the storage pool increases proportionally. For four EC2s the aggregate throughput is about 94 MB/s, roughly four times the throughput for a single configuration.

The total time required to transfer the 1 GB data set to each of the EC2s is around 43 s, close to that required to transfer a 1 GB data set to a single client from a single EBS volume. Hence, CloudMW provides a scalable data sharing architecture for data transfer. Applications can use this facility to rapidly transfer shared data from EBS volumes to local storage in ELDs where they can be accessed at even higher date rates.

The maximum throughput of an EC2 node limits the number of client connections made to it. To scale beyond this limit, one can build a network of EC2 nodes (similar in spirit to the use of a multi-stage interconnection network instead of a crossbar) to build a scalable data transfer network between the EBS volumes to the computing nodes. The tradeoff is an increase in latency for small requests, but the throughput can scale indefinitely.

6 Related work

Currently, cloud computing is a hot research topic and several schemes have been proposed recently for performance optimization of cloud storage. These approaches can be categorized as follows.

Cloud computing performance analysis: Early work on cloud performance was done by Palankar *et al.* (2008). They analyzed the storage performance of Amazon S3 and identified the shortcomings for scientific applications. In addition, Brantner *et*

al. (2008) presented the advantages and challenges of using S3 for database applications and studied the performance, cost, and consistency characteristics. Walker (2008) argued that cloud computing is not suitable for scientific research by comparing it with the traditional high performance clusters. Wang and Ng (2010) have done comprehensive testing of the network performance in Amazon clouds. They have measured several metrics on Amazon EC2 instances, including processor sharing, end-to-end delay, TCP/UDP throughput, and packet loss, and observed restricted processing sharing, unstable bandwidth, and abnormal delay variations. Our ideas were motivated by these results. If the performance variations and fluctuations are not urgently addressed, many exciting new applications will be excluded from using the cloud.

Cloud storage optimization: Amazon Dynamo (DeCandia *et al.*, 2007) demonstrated the fundamental service-oriented architecture for cloud storage. The large difference of our work with Dynamo is the cooperative competition protocol that provides stable performance to facilitate QoS guarantees. Lim *et al.* (2010) proposed a multi-tiered middleware architecture in the Hadoop Distributed File System (HDFS) to rebalance the data requests between the cloud provider and EC2s. However, our work is focused on how to avoid performance fluctuation but not balance the requests over HDFS. This optimization schema complements our work because the I/O scheduler is also necessary in our framework.

Cloud storage security: Security for cloud storage is an important issue that is being actively researched in both industrial and academic fields. How best to achieve data reliability and enforce security for cloud computing is an open problem. OceanStore (Kubiatowicz *et al.*, 2000) was designed to minimize trust for durability over the storage system. In addition, Mahajan *et al.* (2010) designed a cloud storage that minimizes trust assumptions to guarantee safety and protect availability. Meanwhile, it can resist large-scale correlated data failures.

Cloud storage infrastructure: For the general customers, the cloud acts as a black box. It is impossible to modify and optimize the architecture of cloud storage. Companies like Amazon, Google, Microsoft, Netapp, and EMC provide new features for their public or proprietary solutions. Recently, Google has announced a new service, called Google Storage, during its I/O Conference 2010, while Amazon announced a cheaper storage solution, reduced redundancy storage (RRS) (Amazon S3, 2010), which automatically reduces the number of replicas maintained by S3. This dovetails with our strategy of maintaining replicas through the middleware for explicit performance stability.

7 Conclusions and future work

Many organizations have started to use cloud services for different kinds of applications. For example, recently, researchers have been using the cloud to build large-scale high performance decryption testbeds to verify code security. One can imagine that more and more applications will be deployed in the cloud as the performance, security, QoS, and pricing issues get resolved. In this paper, we propose an optimization middleware between the cloud storage provider and the cloud computing layer. We adopt several schemas, including chunking, replication, and cooperative competition, as an approach to addressing the performance fluctuation problem of current cloud computing. Our solution can decrease the possibility of performance variation and pave the way for realistic I/O QoS guarantees. The evaluation results of our framework are promising and indicate that our middleware is able to obtain expected effectiveness. We are continuing to develop the framework to make it more robust and applicable for a larger variety of data intensive computations.

References

Amazon AWS, 2010. Amazon Web Services. Available from http://aws.amazon.com [Accessed on Sept. 21, 2010].

Amazon EBS, 2010. Amazon Elastic Block Store. Available from http://aws.amazon.com/ebs/ [Accessed on Sept. 21, 2010].

Amazon EC2, 2010. Amazon Elastic Compute Cloud. Available from http://aws.amazon.com/ec2 [Accessed on Sept. 21, 2010].

Amazon S3, 2010. Amazon Simple Storage Service. Available from http://aws.amazon.com/s3/ [Accessed on Sept. 21, 2010].

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., *et al.*, 2009. Above the Clouds: a Berkeley View of Cloud Computing. Technical Report, No. UCB/EECS-2009-28, University of California, Berkeley, CA.

Brantner, M., Florescu, D., Graf, D., Kossmann, D., Kraska, T., 2008. Building a Database on S3. Proc. ACM

SIGMOD Int. Conf. on Management of Data, p.251-264. [doi:10.1145/1376616.1376645]

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W., 2007. Dynamo: Amazon's highly available key-value store. *ACM SIGOPS Oper. Syst. Rev.*, **41**(6):205-220. [doi:10.1145/1323293.1294281]

FUSE, 2010. Filesystem in Userspace. Available from http://fuse.sourceforge.net [Accessed on Aug. 21, 2010].

Gulati, A., Merchant, A., Varman, P., 2007. *p*Clock: an Arrival Curve Based Approach for QoS in Shared Storage Systems. Proc. ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems, p.13-24. [doi:10.1145/1254882.1254885]

Gulati, A., Merchant, A., Varman, P., 2010. mClock: Handling Throughput Variability for Hypervisor IO Scheduling. 9th USENIX Symp. on Operating Systems Design and Implementation, p.1-7.

International Data Corporation, 2010. Citing Statistics Information. Available from http://www.idc.com/ [Accessed on Sept. 23, 2010].

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., *et al.*, 2000. OceanStore: an Architecture for Global-Scale Persistent Storage Platforms. Proc. 9th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, p.190-201. [doi:10.1145/378993.379239]

Lim, H.C., Babu, S., Chase, J.S., 2010. Automated Control for Elastic Storage. Proc. 7th Int. Conf. on Autonomic Computing, p.1-10. [doi:10.1145/1809049.1809051]

Mahajan, P., Setty, S., Lee, S., Clement, A., Alvisi, L., Dahlin, M., Walfish, M., 2010. Depot: Cloud Storage with Minimal Trust. 9th USENIX Symp. on Operating Systems Design and Implementation, p.1-12.

Palankar, M.R., Iamnitchi, A., Ripeanu, M., Garfinkel, S., 2008. Amazon S3 for Science Grids: a Viable Solution? Proc. Int. Workshop on Data-Aware Distributed Computing, p.1-9. [doi:10.1145/1383519.1383526]

Postmark, 1997. Postmark: a New File System Benchmark. Available from http://packages.debian.org/stable/utils/postmark [Accessed on Aug. 21, 2010].

VMware ESX, 2010. VMware ESX and ESXi, Bare-Metal Hypervisor for Virtual Machines. Available from http://www.vmware.com/products/esx/ [Accessed on Aug. 21, 2010].

Walker, E., 2008. Benchmarking Amazon EC2 for high-performance scientic computing. *USENIX Log. Mag.*, **33**(5):18-23.

Wang, G.H., Ng, T.S.E., 2010. The Impact of Virtulalization on Network Performance of Amazon EC2 Data Center. INFOCOM, p.1-9. [doi:10.1109/INFCOM.2010.5461931]