Associative Big Data Sharing in Community Clouds: The MeePo Approach

Yongwei Wu, Maomeng Su, and Weiming Zheng, Tsinghua University of China
Kai Hwang, University of Southern California
Albert Y. Zomaya, University of Sydney

MeePo uses associative data sharing, big data metering, data prefetching, privileged access control, and privacy preservation to allow large communities of users to securely share data.

Community clouds, a growing subclass of public clouds,[1,2] appear as a collaborative infrastructure shared by multiple organizations with some common social, research, or business interest.[3,4] Community clouds are often built over datacenters owned by a few organizations. In recent years, these clouds have increased rapidly in the education, business, and government sectors to cope with the growth of big data in these areas.[5]

The service costs to run community clouds are spread over fewer users than in a public cloud. In addition, a higher degree of data sharing is expected in a community cloud than in a private cloud.[2,3] In this article, we describe an associative data sharing method over virtual disks provisioned to serve community user groups. Multiple datacenters work together in colocations of a community cloud. They must handle the rapid growth of data and tolerate disaster or outage in any single datacenter.[4]

We need a unified access model that enables sharing of distributed datasets across multiple datacenters transparently. The sharing of big data is motivated by data dependence or common interests among users.[1,3] Clients can be consumers or producers of the shared data blocks, and the data can be shared by users registered in the same group or associated with different groups. Data sharing could be better protected by privileged accesses to safeguard data integrity and preserve privacy.[6,7]

Traditional datacenters are far from adequate or cost-effective enough to support big data with

 2325-6095/15/$31.00 © 2015 IEEE

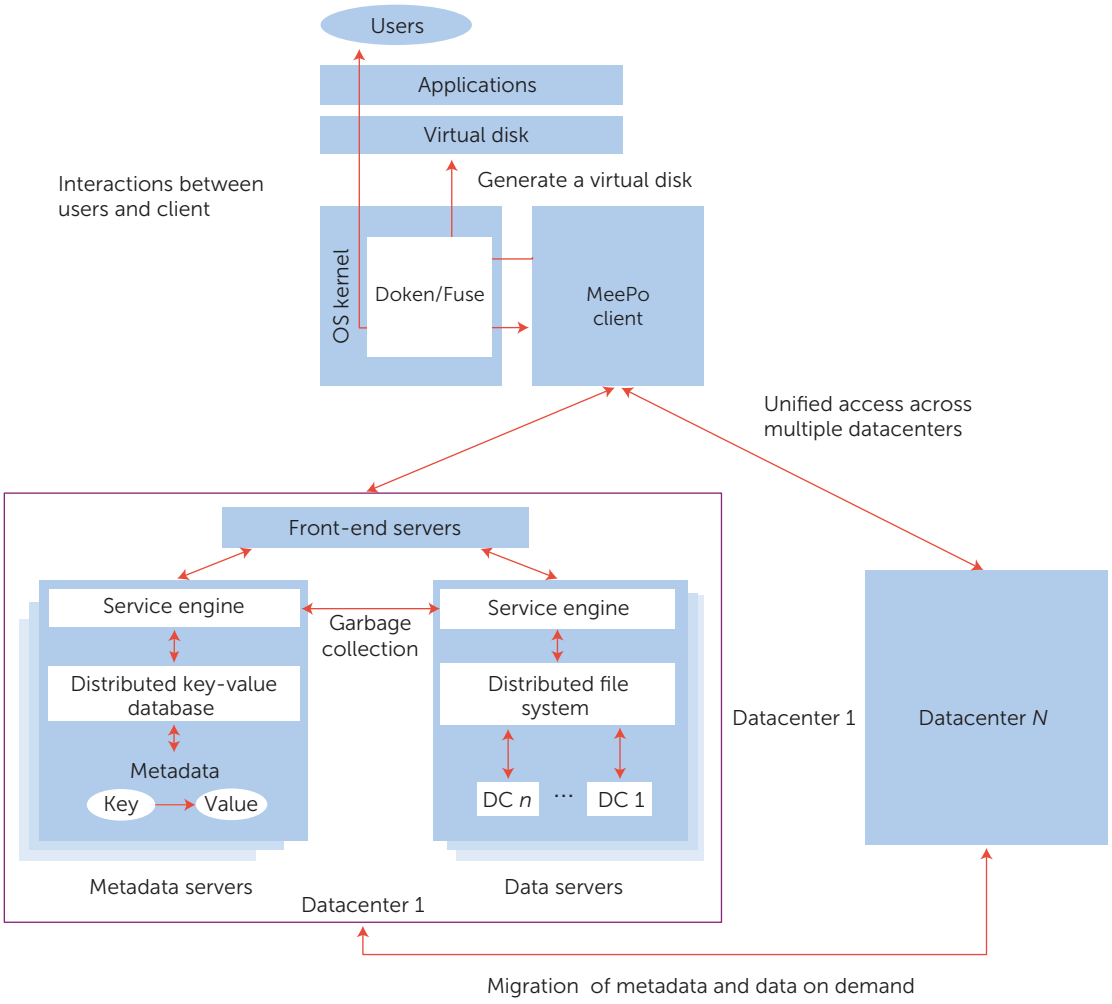

FIGURE 1. MeePo architecture built over multiple interactive datacenters. MeePo is supported by unified data access and data/metadata migration. (DC: data chunk)

guaranteed quality of service (QoS) or performance. Community clouds allow the sharing of big data over virtualized cloud storage to satisfy more users concurrently. Strategies such as that used by Dropbox (www.dropbox.com), which downloads all needed data to a user host before processing it, don't work well in a community cloud, because the datasets involved can be in the terabyte or petabyte range. The storage capacity of local disks is simply inadequate to handle such a large number of data blocks. Virtual disks can alleviate this problem by using elastic resources dynamically. The MeePo architecture is a scalable storage cloud deployed for use by tens of universities and companies in China. MeePo uses a *privileged access control* (PAC) model, which enables the dynamic association of privileged user groups with specific roles to access the shared data blocks.

MeePo Cloud Architecture

Figure 1 shows the MeePo architecture, which is built over multiple datacenters. To support efficient associative data sharing online, MeePo aims to provide scalable storage capacity and low data access latency, prevent outages, and enable disaster recovery. Each datacenter runs independently and supports colocation services. The datacenters communicate with each other through information exchange and data migration. Many clients can log into the datacenters simultaneously.

We built MeePo using commercially available hardware and open source software, and developed cloud software for distributed file management, virtualization support, and user interfaces.

MeePo differs in many aspects from object-oriented storage systems, such as Amazon Simple Storage Service (S3) and Windows Azure, and

COMPARING PUBLIC AND COMMUNITY CLOUDS

We consider only public and community clouds that are converted from interactive datacenters and use broadband connectivity. Table A compares MeePo's technical features and service domains with those of three popular cloud systems.

MySpace (www.myspace.com) acts as a social network service for the general public. It helps users meet people with similar interests, hobbies, or habits, and find popular people to follow. Social network services such as Facebook (www.facebook.com), Twitter (twitter.com), and Instagram (instagram.com) support data sharing using notifications. MeePo, on the other hand, focuses on associative data sharing. It provides virtual disks for users and hierarchically organizes data based on community groups. In this way, data sharing is as simple as operating normal files.

CloudViews, under development at the University of Washington, aims to facilitate communal data sharing in public clouds.[1] It adopts a view abstraction for data sharing, access control, and privacy preservation. However, it only offers a flexible sharing abstraction for public clouds and doesn't support associative big data sharing over community groups and virtual disks, important features that MeePo does incorporate.

Dropbox (www.dropbox.com) provides a storage service for numerous free and paid users globally. It allows users to manually share their data with other users. Dropbox applies encryption and multilevel security based on identity access control. It synchronizes all personal data into the local physical disks from remote datacenters. Box (www.box.com), GoogleDrive (www.google.com/intl/zh-TW/drive), and OneDrive (onedrive.live.com), among others, provide a similar service.

However, these storage services don't support data sharing among several users within a community group. Moreover, associative sharing deals with a large amount of data (terabytes or petabytes). Such a large volume of data is costly to synchronize into the local physical disks. Virtual disks are implemented in MeePo with remote mounting of shared data blocks. MeePo's elastic resources can be extended to support larger amounts of data. Multilevel caching at the client and server sides accelerates data retrieval performance.[2]

References

1. R. Geambasu, S.D. Gribble, and H.M. Levy, "CloudViews: Communal Data Sharing in Public Clouds," *Proc. Workshop Hot Topics in Cloud Computing* (HotCloud), 2009, article 14.
2. D. Beaver et al., "Finding a Needle in Haystack: Facebook's Photo Storage," *Proc. 9th Usenix Symp. Operating Systems Design and Implementation* (OSDI), 2010, pp. 1–8.

Table A. Comparison of public and community clouds.

Cloud provider	Data-sharing mechanisms	Privacy preservation	Access control model	Service domains
Dropbox	Physical disks used	Data encryption, multilevel security	Identity based	Data storage and synchronization service
MySpace	Sharing by notification services	Data encryption	Identity and attribute based	Social networks with data-sharing services
CloudViews	Database-style view abstraction	Signed views	Identity based, view based	Public Web services for data sharing
MeePo	Virtual disks used	Data encryption, third-party key management	Privileged access control	Data-intensive storage, community and social services

software-defined storage systems, such as EMC Elastic Cloud Storage and IBM Spectrum Storage. MeePo acts more like a software-as-a-service (SaaS) platform, oriented for associative and intensive big data sharing based on community groups. MeePo is the only community cloud that leverages virtual disks to share big data blocks (see the sidebar). Strong data encryption is applied with trusted third-party management to secure the keys. MeePo applies the PAC model using both identity and well-

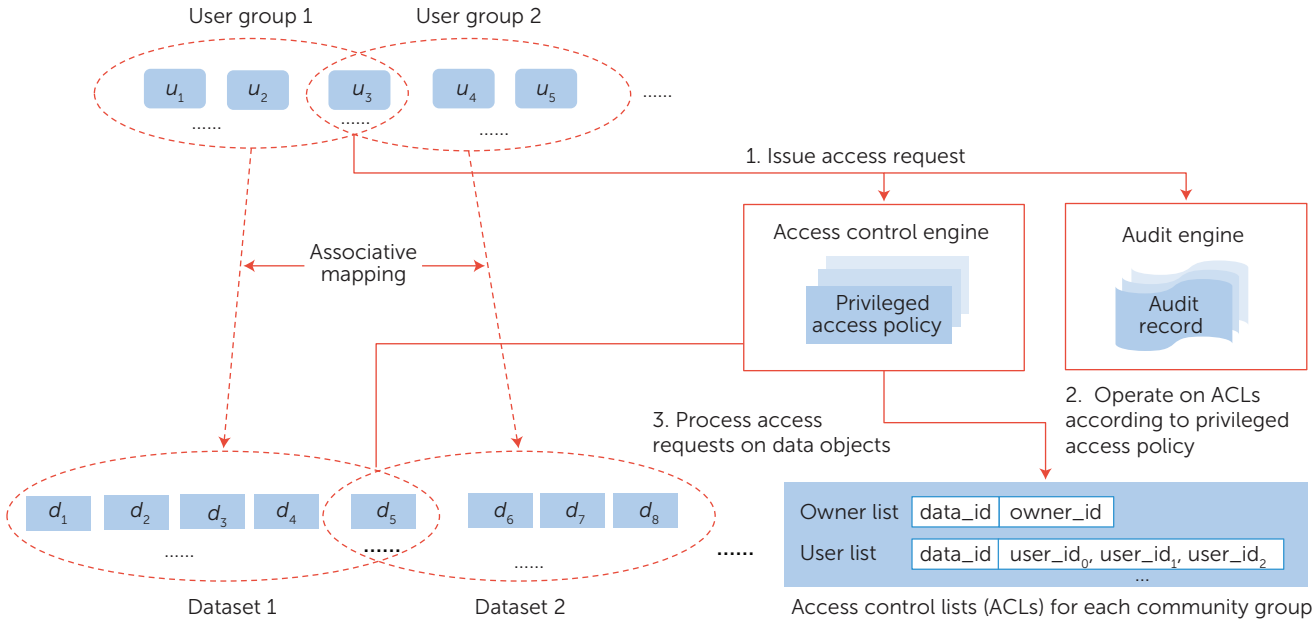

FIGURE 2. Privileged access control (PAC) of shared data blocks by user groups in the MeePo cloud, where *u* and *d* represent individual users and data blocks.

defined user roles in secured access control lists (ACLs).

Front-end servers act as proxies. They receive client requests, route requests to metadata or data servers, and return responses to clients. These servers are stateless and don't keep any metadata or data blocks. During peak hours, additional front-end servers are needed to sustain I/O workloads.

MeePo stores and processes metadata and data in two server clusters. Metadata servers handle user virtual storage quota, access control policies for each data block, and data server locations. The metadata entries are expressed in <key, value> pairs. We use Apache HBase (hbase.apache.org) to build a distributed <key, value> database on the metadata servers. Data servers store the raw data blocks in chunks. We employ the MooseFS (www.moosefs.org) distributed file system to handle blocks distributed on data servers.

The MeePo cloud extends the Dropbox approach using shared virtual disks, which we create using virtualization software. The MeePo client provisions the virtual disks in hosts running Windows, Linux, and Mac OS. We use two software packages to enable the virtual disks: Dokan (dokan-dev.github.io) and Fuse (fuse.sourceforge.net). All data blocks are mapped to common file types in the virtual disks, which are deployed on demand as if the user were creating a data disk on a local computer.

MeePo also supports a virtual disk service for mobile users running Android and iOS applications. The virtual disk enables users to utilize virtual storage, which is offered in unlimited capacity due to dynamic elasticity. Users load the requested files from an identified datacenter. The sharing of data blocks is carried out transparently across all datacenters (see Figure 1).

Metadata and data are frequently migrated among multiple datacenters, which can protect against outage and enable timely disaster recovery.[4] Metadata migration is necessary because the servers maintain a global view of all users and data blocks, which supports unified access across datacenters. Moreover, MeePo replicates some hot data blocks in datacenters to establish a local cache of frequently used blocks. This can reduce access latency by allowing clients to fetch data from the nearest datacenter.

Associative Sharing of Data Blocks

MeePo divides users into community groups. Registered users can create or join groups freely, and can join multiple community groups, but they can only share data with users in the same group. As Figure 2 shows, we define data blocks similarly to Amazon S3's data objects. Datasets are collections of data blocks, each of which can be assigned to multiple groups. Associative sharing implies that all data blocks are dynamically associated or shared by all users in various groups without conflict.

Each group manages its own users and data blocks. MeePo supports both coarse- and fine-grained

Table 1. Access control models for big data storage and sharing.

Use	Identity based (IBAC)	Role based (RBAC)	Lattice based	Attribute based (ABAC)	Purpose based
Cloud systems	Dropbox, Facebook, MeePo, Cisco TrustSec, CloudViews	MeePo, NetWare, Windows NT	Secure Information Flow Systems	MySpace, Access-eGov, VANET	Oracle
Application domains	File storage, Web services, operating systems	Databases, file storage, operating systems	Military and government systems	Web services, databases, and governance	Relational database

data accesses using hierarchically structured directories. As Figure 2 shows, we define coarse-grained data access by mapping user groups to correlated datasets. Support for fine-grained access of data blocks is at the file level for individual users. MeePo maps user groups to all data blocks using hierarchical directories over the virtual disks.

Community versus private directories. Community directories are the highest level at which various user groups can share data blocks within the community. A community directory is often divided into subdirectories for different groups, with each subdirectory handling all data files accessed by users in the associated group. Group-level subdirectories facilitate management of and access to the millions of files that are accessed within a large community.

Associative data sharing involves the creation, uploading, retrieval, modification, and deletion of data blocks. These processes are transformed into the relevant file management operations. Users fetch only the data blocks without downloading the entire database into local hosts. We don't limit the number of data blocks created using the virtual disks. Inactive users can release the virtual disks created, letting other users employ the vacated space. In addition, users can set aside a subdirectory as a private directory, which is inaccessible by other users.

Associative mapping. Each data block appears as a document, image file, or video file. Figure 2 defines an associative mapping between users and data blocks. We denote each user as u_j and each data block as d_k. A user community group G_i is formed by a set of users sharing a common set of data blocks D_l. In MeePo, users can form groups according to their common interests, backgrounds, locations, and so on. Thus, a community group G_i is defined as

$$G_i = \{u_j \mid \text{all } u_j \text{ sharing the same dataset } D_l\}.$$

The groups are essentially data driven, organized from users in government sectors, corporation, school classes, university administrations, family circles, and so on.

Access Control Models

Access control models have been widely studied.[7–10] Table 1 compares five distinct access control models.

- *Identity-based access control* (IBAC) is the most widely used access control model, and is deployed in Dropbox, Facebook, CloudViews, and MeePo.[9]
- The *role-based access control* (RBAC) model has drawn attention in coarse-granular, operating system-specific, and business-specific roles.[8] RBAC appeals to enterprise-wide or cross-enterprise applications where several organizations share roles.
- The *lattice-based* model is typically used in government and military applications. It is used to define the security levels that a data object has and that a user has access to.
- The *attribute-based access control* (ABAC) model is well-suited to big data security and privacy control in Web services.[10] However, malicious users can modify or delete data blocks, which could disable the effective use of shared data by well-behaved users in the same group.
- The *purpose-based* model is mainly used in databases for purpose management. This model combines multiple purposes with each data object, and specifies that some data objects should not be used for certain purposes.

We implemented both fine- and coarse-grained access control models in MeePo to cover the diversity of data types to be accessed. MeePo's PAC model is a combination of IBAC and RBAC.

Privileged Access Control

PAC merges the merits of identity and roles associated with shared data blocks and users. In MeePo,

Input: group_id, user_id, data_owner_id, data_id, user _list, and owner_list
Output: Updated user lists and owner lists

Procedures:
Assign the modify privilege:
1. Check the owner_list by group_id to obtain the owner_id using data_id
2. **if** data_owner_id ! = owner_id **then** the assign right is denied
   / Only the data owner can grant the privilege to modify/
3. **else** get user_list of data_id through group_id and insert user_id into the list

Check the modify privilege:
1. Get owner_list using the group_id and owner_id through the data_id
2. **if** user_id == owner_id **then** the modify right is granted
3. **else** get user_list of data_id through group_id
4. **if** user_id is found in the user_list **then** the modify right is granted
5. **else** the modify request is denied

Check the delete privilege:
1. Get owner_list using the group_id and get owner_id using the data_id
2. **if** user_id ==owner_id **then** the deletion right is granted
3. **else** the deletion request is denied

FIGURE 3. Algorithm 1: Privileged access control of shared data blocks.

only data owners can delete their data blocks. Data owners can authorize select visitors to modify data blocks for their own use. Other users can only read these data blocks. Requests to modify or delete blocks are denied explicitly. MeePo allows users to read or create data blocks freely inside the user groups for sharing purposes. The data blocks are transparently accessed from selected datacenters.

Each user is identified by a unique user_id and each data block by a unique data_id. Access control is implemented using two ACLs for each community group: the *owner* list, which keeps the owner_ids of all data blocks, and the *user list*, for users who gain the privilege to modify certain blocks. These ACLs are managed as <key, value> pairs in the metadata servers for efficient verification.

In using the owner list, we apply the pair <data_id, owner_id>. For privileged use, we apply the pair <data_id, user_id lists> to authorize selected users with the modifying privilege on any data block. For a data block owner, the owner_id is identical to the user_id. When a user issues a request to modify a data block, the modify privilege must be checked against the user list. This access control process is performed at the data block level. The delete privilege can be similarly controlled. Furthermore, an audit engine continuously audits access requests to track all users' actions, including accessing data and granting permissions, or trace events leading up to an error.[11]

Algorithm 1 (Figure 3) specifies the PAC procedure, which we extended from using ACLs on identities. The PAC also classifies users as having different role-based privileges to modify or delete the selected data blocks.

Data Prefetching Policy for Faster Access

In MeePo, the read operation (data downloading) is performed more frequently than the write operation (data uploading). This is because shared data blocks are often repeatedly retrieved after they're created. Consequently, the prefetching policy for the read operations will make the most use of the network bandwidth and hide latency.[12] This action improves the data retrieval performance and enhances the user experience. In random access mode, we don't apply any prefetching policy. In sequential and interleaved access modes, we implement a prefetching method similar to the on-demand read-ahead algorithm reported elsewhere.[13]

Sequential and interleaved access modes account for 81.4 percent of MeePo's total data traffic with a higher read frequency (80.5 percent). As a result, the prefetching policy brings the most benefits to read operations. The prefetching size plays an important role in network I/O performance and hit

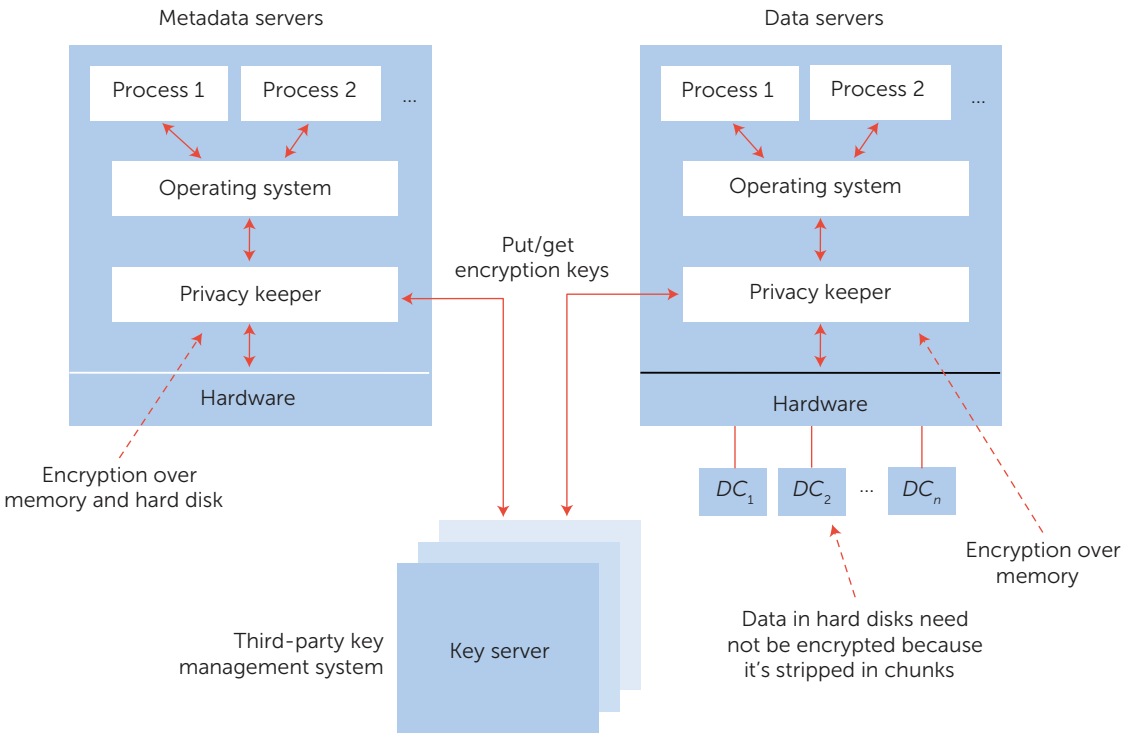

FIGURE 4. Data privacy protection mechanisms built into the MeePo cloud. MeePo uses three types of servers to handle the raw data, metadata, and key management. (DC: data chunks)

ratio on virtual disks. If a request changes the offset to read and misses in the window, this window will be closed and a new prefetching operation will take over.

Data Privacy Protection

We implemented some data protection measures inside the datacenters, as Figure 4 illustrates. To prevent malicious administrators with root access authority (in particular, to the operating system platform[14]) from abusing or compromising data, we designed a *privacy keeper*, a virtual-machine-based mechanism to guard cloud resources. We encrypt the data to be stored in memory and disks.

When authorized processes issue requests to access their own data, the privacy keeper retrieves the data from memories or disks. The privacy keeper then decrypts the data and returns it to the processes in plaintext. The privacy keeper guarantees the isolation of each process. Unlike the management data in metadata servers, which is protected in both memory and disks, data servers only need guarding from the memory as the raw data is stored in chunks. Reconstructing the original data from a few chunks is difficult without the metadata.

Each chunk is replicated (usually three copies) in case of disaster or corruption. Moreover, all client requests are transferred using HTTPS to prevent it from being captured by unauthorized users. This method ensures both data privacy and data integrity. Encryption key management is one of the most important issues a system should address to protect its data. In MeePo, we rely on a trusted third-party key management system deployed in the key servers.

Big Data Profiling and Measured Performance

We deployed MeePo services at Tsinghua University (THU) in early 2012. Users from about a dozen universities and companies in China quickly joined. Currently, almost 30,000 users are registered, forming more than 600 community groups, and sharing 3 million data blocks of various sizes. In total, more than 150 Tbytes of data are stored in two datacenters on the THU campus. Presently, popular cloud applications built on MeePo include data banks, entertainment, a healthcare cloud, dating services, and dancing, swimming, and skating clubs.

Figure 5a shows the growth of data traffic on MeePo in 2014. The amount of data (including new and modified data) uploaded to MeePo is rather stable at 7.5 Tbytes per month. The data retrieved from MeePo increased to about 160 Tbytes within a year, implying that data sharing among users increased sharply. In other words, users perform read opera-

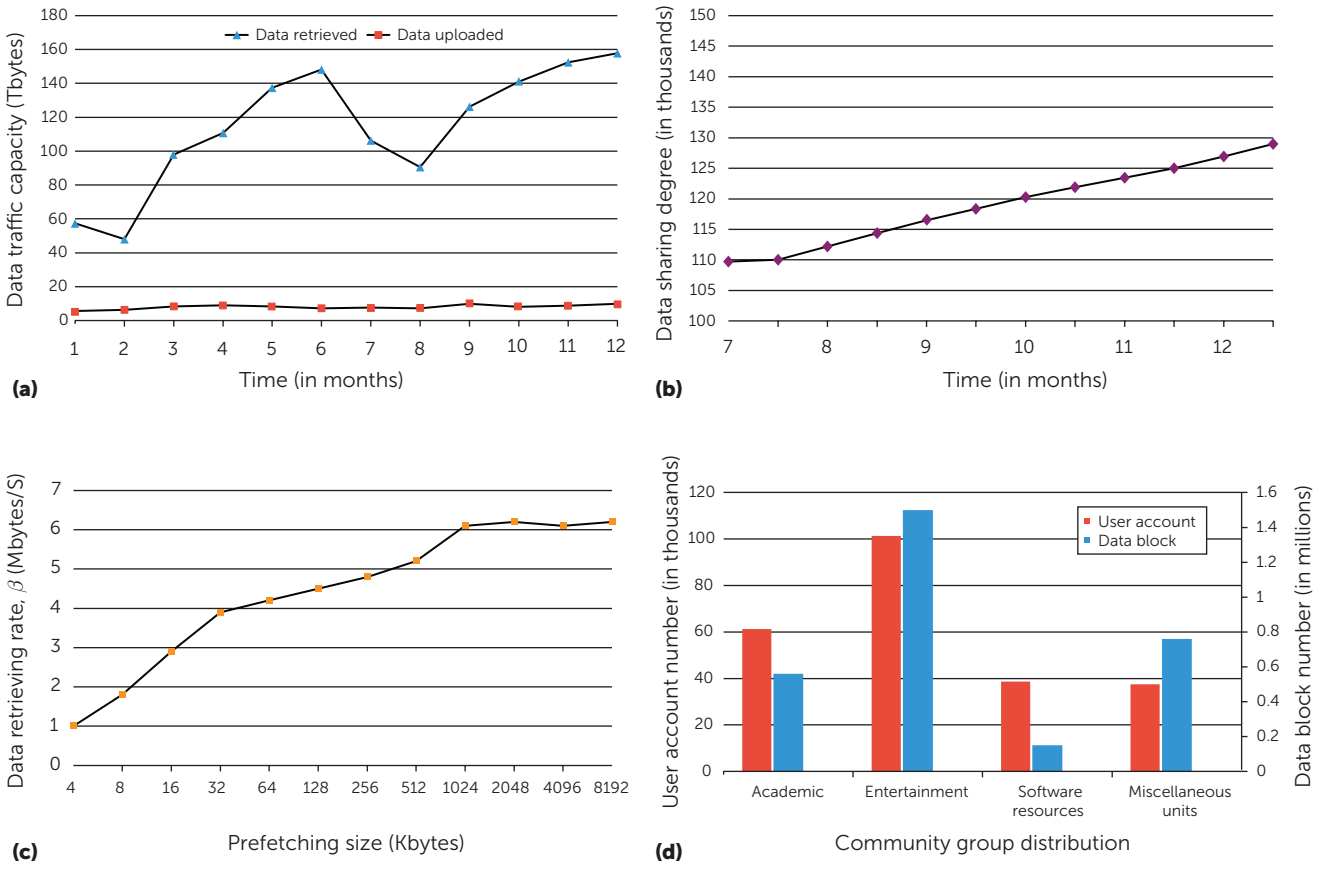

FIGURE 5. Big data profiling and performance results on MeePo cloud measured at Tsinghua University through 2014: (a) data growth rate, (b) data sharing degree, (c) data retrieval rate, and (d) community group distribution.

tions much more frequently than write operations. This justifies the need for data prefetching to reduce the latency and response time.

To reflect the benefits of data sharing, we define the data sharing degree, denoted by α, as the number of data blocks shared among users on average. The larger α is, the larger the number of data blocks each user can utilize and the greater the benefits users will obtain from data sharing. Let S_i be the number of users in a group, T_i be the number of data blocks in the same group, and M be the total number of users in MeePo. In addition, let N be the total number of groups. Because users can join more than one group, M is the upper bound by the sum of S_i of these N groups. Hence, we can define α as

$$\alpha = \sum_{i=1}^{N} (S_i \times T_i) / M .$$

Figure 5b plots the data sharing degree for 2014. The average number of data blocks shared by users increased linearly, reaching about 130,000 in late 2014. This measurement shows that all users can benefit from a high degree of data sharing.

The prefetching size has a direct impact on network I/O performance. The retrieval rate is impacted by the prefetching size, as Figure 5c shows. We define data retrieval rate β as the ratio of D to T ($\beta = D/T$), where D is the size of a single data block retrieved and T is the total time it takes to transfer the block from the cloud to the local hosts.

Figure 5c plots the data retrieval rate under different prefetching sizes. The data retrieval rate improves as the prefetching size increases and is saturated at 1 Mbyte. The network I/O bandwidth limits the prefetching size. This rate results in a 6 Mbytes per second (Mbps) data read rate to facilitate high-speed big data sharing, almost two times faster than the 2.95 Mbps read rate without any data prefetching. The write data rate is much lower, between 1.8 and 2.5 Mbps.

We divide community groups in MeePo into four categories of services, as Figure 5d shows:

- *Academic* includes groups created for research and education.
- *Entertainment* enables user groups to share video, music, computer games, and so on.
- *Software* groups are motivated by uploading and retrieving software packages.
- *Miscellaneous* units are comprised of social groups.

As Figure 5d shows, there were more than 100,000 user accounts and about 150,000 data blocks in the entertainment category, demonstrating that associative data sharing encourages more users to share big data. Because users can join multiple community groups, the number of user accounts collected in Figure 5d is much greater than the number of actual MeePo users.

The MeePo cloud design makes several contributions to the field. Of particular importance is its use of virtual disks and metadata servers to handle sharing-intensive big data. We attempt to exploit the built-in elastic storage resources in community clouds. In addition, because MeePo uses the PAC model, which combines the advantages of IBAC and RBAC models, it supports low-cost community cloud services. Its use of privacy checkers in both data and metadata servers enables tight enforcement of data privacy policies,[14] and its use of local key servers supports strong key management in community clouds without resorting to the use of expensive PKI services.

One shortcoming in current MeePo operations is the need to upgrade the simple PAC model to serve multiple organizations in community cloud services. MeePo's cloud security structure should be extended from a single level to multiple levels, hierarchically. We also need autotiering solutions or data coloring solutions in a more secure storage cloud platform.[6]

We can further the security of data creation, transferring, and sharing processes using strong authentication, privileged access authorization, and user accountability. These approaches could be even more improved with a reputation-based trust management system.[6] Reputation systems can be used to enforce service-level agreements and support real-time security/compliance monitory and data provenance to uphold data integrity. A discussion of extended performance issues of public clouds is available elsewhere.[15]

Acknowledgments

This work is supported by the National High-Tech R&D (863) Program of China (2013AA01A213), Natural Science Foundation of China (61433008, 61373145, 61170210, U1435216), and Chinese Special Project of Science and Technology (2013zx01039-002-002). Kai Hwang acknowledges support from China's 973 Basic Research Grant 2011CB302505, Guangdong Innovation Team Grant201001D0104726115, and EMC funding support for an endowed visiting professorship at Tsinghua University.

References

1. S. Goyal, "Public vs Private vs Hybrid vs Community—Cloud Computing: A Critical Review," *Int'l J. Computer Network and Information Security*, vol. 6, no. 3, 2014, pp. 20–29.
2. M. Patel, R. Patel, and A.R. Chaube, "Hybrid Cloud Computing Data Sharing & Security Issues," *Int'l J. Research*, vol. 2, no. 1, 2015, pp. 464–467.
3. R. Geambasu, S.D. Gribble, and H.M. Levy, "CloudViews: Communal Data Sharing in Public Clouds," *Proc. Workshop Hot Topics in Cloud Computing* (HotCloud), 2009, article 14.
4. K. Hwang, G. Fox, and J. Dongarra, *Distributed and Cloud Computing*, Morgan Kauffann, 2012.
5. K. Kambatla et al., "Trends in Big Data Analytics," *J. Parallel and Distributed Computing*, vol. 74, no. 7, 2014, pp. 2561–2573.
6. K. Hwang and D. Li, "Trusted Cloud Computing with Secured Resources and Data Coloring," *IEEE Internet Computing*, vol. 14, no. 5, 2010, pp. 14–22.
7. S. Yu et al., "Achieving Secure, Scalable, and Fine-Grained Access Control in Cloud Computing," *Proc. IEEE Infocom*, 2010, pp. 534–542.
8. E. Barka, S.S. Mathew, and Y. Atif, "Securing the Web of Things with Role-Based Access Control," *Codes, Cryptology, and Information Security*, LNCS 9084, Springer, 2015, pp. 14–26.
9. N. Saxena, G. Tsudik, and J. Yi, "Identity-Based Access Control for Ad Hoc Groups," *Int'l Conf. Information Security and Cryptology*, LNCS 3506, Springer, 2004, pp. 362–379.
10. V.C. Hu et al., *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*, NIST Special Publication 800-162, 2014; http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf.
11. M. Sookhak et al., "Remote Data Auditing in Cloud Computing Environments: A Survey, Taxonomy, and Open Issues," *ACM Computing Surveys* (CSUR), vol. 47, no. 4, 2015, article 65.
12. D. Beaver et al., "Finding a Needle in Haystack: Facebook's Photo Storage," *Proc. 9th Usenix*

Symp. Operating Systems Design and Implementation (OSDI), 2010, pp. 1–8.
13. Q. Wang et al., "Improving the Effective IO Throughput by Adaptive Read-Ahead Strategy for Private Cloud Storage Service," *Proc. China Grid Annual Conf.*, 2012, pp. 134–141.
14. F. Zhang et al., "CloudVisor: Retrofitting Protection of Virtual Machines in Multi-tenant Cloud with Nested Virtualization," *Proc. ACM Symp. Operating Systems Principles* (SOSP), 2011, pp. 203–216.
15. K. Hwang et al., "Cloud Performance Modeling with Benchmark Evaluation of Scaling Strategies in Elastic Clouds," *IEEE Trans. Parallel and Distributed Systems*, forthcoming.

YONGWEI WU *is a professor of computer science and technology at Tsinghua University, Beijing. His research interests include parallel and distributed processing and systems and cloud computing. Wu has a PhD in applied mathematics from the Chinese Academy of Sciences. He's a member of IEEE. Contact him at wuyw@tsinghua.edu.cn.*

MAOMENG SU *is a PhD candidate in the Department of Computer Science and Technology at Tsinghua University, Beijing. His research interests include cloud storage systems, distributed systems, new datacenter networks, remote direct memory access, and key-value store systems. Su has a BS in engineering from the University of Science and Technology Beijing. Contact him at smm11@mails.tsinghua.edu.cn.*

WEIMIN ZHENG *is a professor of computer science and technology at Tsinghua University, Beijing, where he's also the research director of the Institute of High Performance Computing and the managing director of the Chinese Computer Society. His research interests include computer architecture, operating system, storage networks, and distributed computing. Zheng has an MS in computer science and technology from Tsinghua University. He's a member of IEEE. Contact him at zwm-dcs@tsinghua.edu.cn.*

KAI HWANG *is a professor of electrical engineering and computer science at the University of Southern California. His research interests include computer architecture, parallel processing, distributed systems, high-performance computing, and cloud computing. Hwang has a PhD in electrical engineering and computer science from the University of California, Berkeley. He's a life fellow of the IEEE Computer Society. Contact him at kaihwang@usc.edu.*

ALBERT Y. ZOMAYA *is the Chair Professor of High Performance Computing & Networking in the School of Information Technologies at the University of Sydney, and the director of the Centre for Distributed and High Performance Computing. His research interests are in the areas of parallel and distributed computing and complex systems. Zomaya has a PhD in automatic control and systems engineering from Sheffield University, UK. He's a chartered engineer and a Fellow of the American Association for the Advancement of Science, IEEE, and the Institution of Engineering and Technology. Contact him at albert.zomaya@sydney.edu.au.*