Data Retrieval and Privacy Enhancing Hybrid P2P-Cloud Storage System

Gi Seok Park [*] and Hwangjun Song [†]
[*] Division of IT Convergence Engineering, Pohang University of Science and Technology, Pohang, Republic of Korea.
E-mail: kiseok@postech.ac.kr Tel: + 82 (54)-279-5684
[†] Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang, Republic of Korea.
E-mail: hwangjun@postech.ac.kr Tel: + 82 (54)-279-2246

Abstract— **In this work, we present a novel fountain code-based hybrid storage system that combines cloud storage with P2P storage. In the cloud storage system, all the data are consistently stored at server clusters but the data may be exposed to others accessing server clusters. On the other hand, in the P2P storage system, the data are distributed to a group of participating peers but the data retrieval may be significantly degraded. The proposed hybrid system is designed to provide the desired data retrieval with a short upload time and maintain the privacy to avoid other users from reading the contents.**

Keywords—hybrid P2P-cloud storage; fountain codes; data privacy; hybrid storage system stability; data retrieval

I. Introduction

Many storage services such as Amazon Glacier, Google Drive, and Microsoft SkyDrive have already been successfully deployed over the Internet. In general, remote storage systems can be classified into two groups: cloud storage systems and P2P (peer to peer) storage systems. In cloud storage systems, a high level of data retrieval can be guaranteed using a replication technique for data backup. In this case, data retrieval means the probability that the user can obtain his/her data successfully. However, private data may be exposed to other users since all data is stored at the server. Therefore, data privacy is one of the most important issues for cloud storage systems. On the other hand, in P2P storage systems, some of the problems that are inherent in cloud storage can be solved. A unique characteristic of the P2P system is that peers share their resources with each other when they join a network. Therefore, as more peers participate in the system, P2P storage can be extended constantly. In P2P storage, data privacy can also be improved because data is divided into blocks and distributed among a group of participating peers. However, the relatively low data retrieval is a serious obstacle to the successful deployment of P2P storage. Thus, data retrieval is considered as one of the most important performance measures for P2P storage systems. Much research effort has been devoted to provide data retrieval equal to that of cloud storage systems [2]. Until now, the most efficient method for achieving high data retrieval for P2P storage is to use erasure protection codes and to store encoded data in multiple peers until the required data retrieval is satisfied. However, this may increase the upload time. Thus, it is very important to reduce transmission time for the successful deployment of P2P storage.

In this work, we propose a fountain codes-based hybrid P2P-cloud storage system to provide data privacy, storage scalability, and data retrieval. Some of the unique features of the proposed algorithm are that it adopts rateless fountain codes and includes a packet distribution algorithm in order to guarantee the required level of data retrieval and privacy. Furthermore, data is stored with a short upload time. The rest of the paper is organized as follows. The proposed hybrid P2P-cloud storage system is then presented in Section II. Simulation results are provided in Section III. Finally, concluding remarks are presented in Section IV.

II. Proposed Hybrid P2P-Cloud Storage System

The main goal of the proposed hybrid P2P-cloud storage system is to minimize the upload time while satisfying the required data retrieval and supporting the privacy of user data stored on P2P and cloud storage system. The proposed hybrid storage system adopts the usage of fountain codes to achieve our goal. Thanks to the generic characteristics of fountain codes, we can solve the invasion of data privacy by controlling the number of encoded symbols stored on the cloud storage and peers in P2P storage and improve data retrieval by distributing sufficient number of encoded symbols to the cloud storage and peers. Before presenting the proposed hybrid storage system in detail, we make the following definitions:

- **Cloud storage and peer accessibility** means the probability that the user is able to access the data stored on cloud storage and peers during the given time interval.
- **System stability** denotes the probability that the user retrieves more than the minimum number of encoded symbols required for successful fountain decoding.
- **Data retrieval** stands for the probability that user obtains his/her own data successfully without any errors.

A. System Architecture

The proposed hybrid storage system architecture is shown in Figure 1. The proposed system consists of four main components: a node manager, a packet distributor, a fountain encoder, and a transmission scheduler. The node manager receives the information of the server in cloud storage and peers in P2P storage from the bootstrap server, and estimates the bandwidth available with them. The packet distributor determines the packet distribution vector (i.e. how much encoded data should be stored on the server and peers) and the code rate of the fountain encoder based on the feedback information. The fountain encoder divides a file into several blocks, and generates encoded symbols for every source block at the code rate determined by the parameter control unit. The transmission scheduler allocates encoded symbols to each node based on the estimated bandwidth and packet distribution vector until the ACK message arrives. In the proposed system, the FFDH (first-fit decreasing height) algorithm [4], which provides a near optimal solution for the scheduler design problem, is employed for an transmission scheduler.

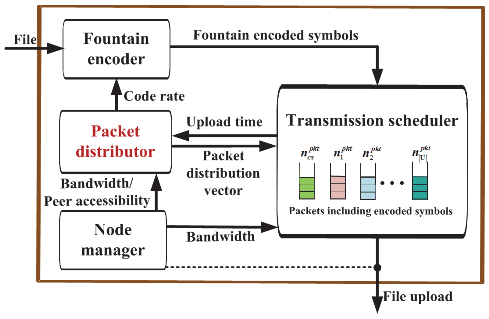
File
Fountain encoder
Fountain encoded symbols
Code rate
Packet distributor
Upload time
Packet distribution vector
Transmission scheduler
n_{cs}^{pkt} n_1^{pkt} n_2^{pkt} $n_{|U|}^{pkt}$
Packets including encoded symbols
Bandwidth/ Peer accessibility
Node manager
Bandwidth
File upload

Fig. 1. Architecture of the proposed hybrid P2P-cloud storage system.

B. Problem Description

In the proposed system, our control variables are c and packet distribution vector $\vec{n}_{pkt} = \left(n_{cs}^{pkt}, n_1^{pkt}, \cdots, n_i^{pkt}, \cdots, n_{|\mathbf{U}_{initial\_set}|}^{pkt} \right)$, where n_{cs}^{pkt} and n_i^{pkt} denotes the number of packets including the encoded symbols that the user allocates in cloud storage and the i_{th} peer over P2P storage, respectively. $\mathbf{U}_{initial\_set}$ is the peer set provided by the bootstrap server. At this point, we need to describe the hybrid storage system stability and data retrieval because the fountain decoding process may fail with a very low probability even though $M_{\min}$ encoded symbols are accessible, where $M_{\min}$ is the minimum number of encoded symbols required for successful decoding.

Peer accessibility: In an actual P2P system, it is observed that the peer lifetime depends on the time it stays in the system [5]. Thus peer accessibility is defined as follows:

$$p_i = P\left\{ X \geq T_i^{stay} + T_{req} \middle| X \geq T_i^{stay} \right\} = \frac{P\left\{ X \geq T_i^{stay} + T_{req} \right\}}{P\left\{ X \geq T_i^{stay} \right\}}, \qquad (2)$$

where a random variable X represents the lifetime of a peer. Eq. (2) gives the probability that a peer who has stayed for T_i^{stay} will survive longer than T_{req} in the system. Hence, the peer lifetime should be modeled by a probability distribution with a memory property that remembers T_i^{stay}. In our proposed system, a Pareto distribution is adopted for modeling the lifetime of a peer [5].

Hybrid P2P-cloud storage system stability: The hybrid storage system stability is calculated based on $\vec{p} = \left(p_{cs}, p_1, \cdots, p_i, \cdots, p_{|\mathbf{U}_{initial\_set}|} \right)$, where p_{cs} is server accessibility. Two vector matrices are defined. The node matrix is characterized by

$$\mathbf{Q} = \left(\vec{q}_1 \quad \vec{q}_2 \quad \cdots \quad \vec{q}_i \quad \cdots \quad \vec{q}_{N_{row}} \right)^{\mathrm{T}},$$
$$\vec{q}_i = \left(q_{i,cs}, q_{i,1}, q_{i,2}, \cdots, q_{i,j}, \cdots, q_{i,|\mathbf{U}_{initial\_set}|} \right),$$
$$N_{row} = 2 \cdot \sum_{x=1}^{|\mathbf{U}_{initial\_set}|} \binom{|\mathbf{U}_{initial\_set}|}{x}. \qquad (3)$$

In the above equation, the corresponding element in the available node status vector $\vec{q}_i$ is set to one if the cloud storage or peers are consistently available in the system for the time T_{req}; otherwise, it is fixed as zero. The event matrix can be characterized by

$$\mathbf{V} = \left(\vec{v}_1 \quad \vec{v}_2 \quad \cdots \quad \vec{v}_i \quad \cdots \quad \vec{v}_{N_{row}} \right)^{\mathrm{T}},$$
$$\vec{v}_i = \left(v_{i,cs}\left(\vec{n}_{pkt}\right), v_{i,1}\left(\vec{n}_{pkt}\right), \cdots, v_{i,j}\left(\vec{n}_{pkt}\right), \cdots, v_{i,|\mathbf{U}_{initial\_set}|}\left(\vec{n}_{pkt}\right) \right),$$
$$v_{i,cs}\left(\vec{n}_{pkt}\right) = \begin{cases} p_{cs} & \text{if } N_{ps} \cdot \left(\vec{q}_i \bullet \vec{n}_{pkt} \right) \geq M_{\min} \text{ and } q_{i,cs} = 1 \\ 1 - p_{cs} & \text{if } N_{ps} \cdot \left(\vec{q}_i \bullet \vec{n}_{pkt} \right) \geq M_{\min} \text{ and } q_{i,cs} = 0, \\ 0 & \text{otherwise} \end{cases} \qquad (4)$$

$$v_{i,j}\left(\vec{n}_{pkt}\right) = \begin{cases} p_j & \text{if } N_{ps} \cdot \left(\vec{q}_i \bullet \vec{n}_{pkt} \right) \geq M_{\min} \text{ and } q_{i,j} = 1 \\ 1 - p_j & \text{if } N_{ps} \cdot \left(\vec{q}_i \bullet \vec{n}_{pkt} \right) \geq M_{\min} \text{ and } q_{i,j} = 0, \\ 0 & \text{otherwise} \end{cases} \qquad (5)$$

where $\bullet$ denotes the inner product and N_{ps} is the number of encoded symbols in a packet. The multiplication of all components in $\vec{e}_i$ denotes the probability that more than $M_{\min}$ encoded symbols can be obtained when the accessible node status vector is $\vec{q}_i$. Now, $SS\left(\vec{n}_{pkt}\right)$ is represented by

$$SS\left(\vec{n}_{pkt}\right) = \sum_{i=1}^{N_{row}} \left(v_{i,cs}\left(\vec{n}_{pkt}\right) \cdot \prod_{j=1}^{|\mathbf{U}_{initial\_set}|} v_{i,j}\left(\vec{n}_{pkt}\right) \right). \qquad (6)$$

Data retrieval: As mentioned earlier, the fountain decoding process may fail even though $M_{\min}$ encoded symbols are obtained. In [6], the relationship among $\delta_{failure}$, $M_{\min}$, and M_{source} is found as follows.

$$M_{\min} = M_{source} + 2 \cdot \ln\left(\frac{\omega \cdot \ln\left(\frac{M_{source}}{\delta_{failure}} \right) \cdot \sqrt{M_{source}}}{\delta_{failure}} \right) \cdot \omega \cdot \ln\left(\frac{M_{source}}{\delta_{failure}} \right) \cdot \sqrt{M_{source}}, \qquad (7)$$

Where M_{source} is the number of source symbols and ω is an extra parameter of the robust soliton distribution with a value less than one. Consequently, in this work, data retrieval is defined by combining Eqs. (6) and (7).

$$R\left(\vec{n}_{pkt}\right) = \left(1 - \delta_{failure}\right) \cdot SS\left(\vec{n}_{pkt}\right). \qquad (8)$$

The above $R\left(\vec{n}_{pkt}\right)$ is the probability to obtain M_{source} source symbols successfully from the retrieved encoded symbols. Now, we can formulate the problem as follows.

Problem formulation: Determine $\vec{n}_{pkt}$ to minimize the total upload time

$$T_{up}\left(\vec{n}_{pkt}, \overrightarrow{bw}\right), \tag{9}$$

$$\text{subject to } R\left(\vec{n}_{pkt}\right) \geq P_{retrival}^{\min}, \tag{10}$$

$$n_{cs}^{pkt} < \left\lceil \frac{M_{source}}{N_{ps}} \right\rceil, \text{ and} \tag{11}$$

$$n_{i}^{pkt} < \min\left\{ \left\lceil \frac{h_i}{S_{symbol} \cdot N_{ps}} \right\rceil, \left\lceil \frac{M_{source}}{N_{ps}} \right\rceil \right\}, for \quad 1 \leq i \leq \left|\mathbf{U}_{initial\_set}\right|, \tag{12}$$

where Eqs. (11) and (12) are included in order to avoid the invasion of data privacy since the decoding process is not performed successfully with less than M_{source} encoded symbols. $\overrightarrow{bw} = \left(bw_{cs}, bw_1, \cdots, bw_i, \cdots, bw_{\left|\mathbf{U}_{initial\_set}\right|}\right)$ is the available bandwidth vector, where bw_{cs} and bw_i indicate the estimated available bandwidth with server and i_{th} peer, respectively. $P_{retrival}^{\min}$ is the minimum required data retrieval and h_i is the remaining storage space in the i_{th} peer.

C. *Proposed Packet Distribution Algirithm*

The proposed algorithm determines the initial peer set and the packet distribution vector with low computational complexity. First, we have to determine the initial peer set provided by the bootstrap server. Whenever a service request is received, the bootstrap server has to provide the user with the peer set in which a feasible solution to the given problem exists. If the constraint in Eqs. (10) and (12) cannot be satisfied with the given initial peer set, the user requests additional peers until an effective solution is obtained.

Next, we control the number of encoded symbols stored on server and each peer to minimize the upload time while satisfying data retrieval and privacy constraints. Basically, we adopt the Branch and Bound algorithm (B&B) to obtain the optimal solution of the given problem with a low computational complexity. However, this still requires a considerable amount of computation. Thus, we propose the Branch and Bound algorithm with Dynamic Step Size (B&B with DSS) for the fast convergence: This algorithm addresses how to obtain a near optimal solution for $\vec{n}_{pkt}$ with low computational complexity compared to the traditional B&B algorithm. To determine the number of packets including the encoded symbols transmitted to cloud storage and peers, the packet distribution algorithm is summarized as follows.

Step 0: First, set the number of intervals N_{int} to 2^{α}, where α is an integer between 0 and $\left\lfloor \log_2 {M_{source}}/{N_{ps}} \right\rfloor$. For $\forall i \; \left(0 \leq i \leq \left|\mathbf{U}_{initial\_set}\right|\right)$, initialize $n_i^{\min} = 0$, $n_i^{\max} = \left\lfloor \log_2 {M_{source}}/{N_{ps}} \right\rfloor$, and $N_i^{\text{int}} = N_{\text{int}}$. Then, calculate the interval points using the equation

$$n_i^{pkt}\left(k\right) = n_i^{\min} + k \cdot \frac{n_i^{\max} - n_i^{\min}}{N_i^{\text{int}}} \quad \text{for } 0 \leq k \leq N_i^{\text{int}}, \tag{13}$$

where $n_i^{pkt}\left(k\right)$ represents the number of packets that are transmitted to the i_{th} peer and k is integer value. The exception case, $n_0^{pkt}\left(k\right)$, represents the number of packets that are assigned to the cloud storage.

Step 1: Calculate the data retrieval and the upload time for all possible combinations of the packet distribution vector determined by Eq. (13) when the FFDH scheduler is adopted. If any combination does not satisfy the constraint in Eq. (10), request for additional peer information from the bootstrap server, and then go back to Step 0.

Step 2: When $\vec{n}_{pkt}^{cur} = \left(n_0^{cur}, n_1^{cur}, \ldots, n_i^{cur}, \ldots, n_{\left|\mathbf{U}_{initial\_set}\right|}^{cur}\right)$ is the packet distribution vector with the minimum upload time that satisfies data retrieval among all the possible combinations in Step 1, the minimum point $n_i^{\min}$ and the maximum point $n_i^{\max}$ are updated by $\max\left\{0,\; n_i^{cur} - \frac{n_i^{\max} - n_i^{\min}}{2 \cdot N_i^{\text{int}}}\right\}$ and $\min\left\{n_i^{cur} + \frac{n_i^{\max} - n_i^{\min}}{2 \cdot N_i^{\text{int}}},\; 2^{\left\lfloor \log_2 {M_{source}}/{N_{ps}} \right\rfloor}\right\}$, respectively. If n_i^{cur} is 0 or $2^{\left\lfloor \log_2 \frac{M_{source}}{N_{ps}} \right\rfloor}$, then N_i^{int} is set to 1; otherwise, N_i^{int} is set to 2.

Step 3: Repeat Steps 1~2 until $n_i^{pkt}\left(k\right) - n_i^{pkt}\left(k-1\right) = 1$.

Step 4: The packet distribution vector $\vec{n}_{pkt}^{cur}$ is chosen to transmit the source blocks.

III. Simulation Results

In general, commercial cloud storage systems are designed for 99.999% stability over a year, which means that the average loss rate of stored data is 0.001% [7]. During the simulation, the cloud storage accessibility is set to 99.9999% to verify the data retrieval enhancement. The peer accessibility is set up taking into consideration the accessibility of peers having relatively long lifetimes since a few peers have a lifetime of months, or even years, in P2P systems [1]. The peer bandwidth is set up based on the up-to-date OECD broadband statistics since all peers use different access networks. As mentioned earlier, to support data privacy, the number of encoded symbols stored on cloud storage and on each peer is limited to less than the number of source symbols. It is assumed that the bootstrap server provides the user with the information of the cloud storage server, 10 peers and 50 additional candidate peers. M_{source} and $M_{\min}$ are determined to 1056 and 1120 by setting $\delta_{failure}$ to 0.00000029 and ω to 0.0025 [6]. The symbol size and the packet payload sizes are set to 32 and 1024 bytes taking into consideration the fountain coding complexity and the amount of overhead. The uplink capacity of the user is set to 100 Mbps and source block size is set to 33 KB.

A. Performance Comparison with Other Algorithms

We compare the proposed algorithm with existing algorithms: the LTCS (LT code-based secure cloud storage service) algorithm [2] and the adaptive ERC algorithm [1]. Since these existing algorithms are proposed for P2P or cloud storage systems, they are simply modified for the purpose of conducting a fair comparison with the proposed hybrid storage system. The existing algorithms are briefly summarized below.

- **LTCS algorithm:** To recover all the source symbols from the distributed encoded symbols that can be retrieved from any l combination of peers, every peer stores at least $M_{\min}/l$ encoded symbols. The source block size is set to 33KB. $\left\lceil M_{source}/N_{ps} \right\rceil - 1$ encoded packets are stored on the cloud storage system to support data privacy.
- **Adaptive ERC algorithm** The appropriate fragment size of an RS code is adaptively chosen to efficiently store data in P2P storage. Additionally, the number of peers that store coded fragments is calculated via a binomial distribution. We choose the most suitable four fragments for a source block size of 33 KB, where any of the four distinctive coded fragments is able to decode the original data. Three fragments are stored on cloud storage to support data privacy.
- **Replication-based redundancy algorithm:** the cloud storage and peers are chosen appropriately and then replicated data is stored on the cloud storage and the selected peers until the data retrieval is satisfied.

The upload time and the amount of redundant data are measured to provide a comparison of the performance with that of existing algorithms. Table I shows the initial network conditions and the accessibility of peers. The accessibility and initial bandwidth of server are 99.999% and 98.381 Mbps, respectively.

TABLE I. ACCESSIBILITY AND BANDWIDTH OF PEERS

NODES	ACCESSIBILITY [%]	BANDWIDTH [Mbps]	NODES	ACCESSIBILITY [%]	BANDWIDTH [Mbps]
PEER 1	90.2	43.392	PEER 6	87.4	95.937
PEER 2	93.3	14.366	PEER 7	89.9	16.287
PEER 3	92.0	15.753	PEER 8	92.5	15.617
PEER 4	90.5	16.550	PEER 9	89.9	16.991
PEER 5	90.4	15.098	PEER 10	92.3	15.474

As shown in Figure 2, the proposed algorithm exhibits the best performance with respect to upload time and storage efficiency except for replication algorithm. When the replication-based redundancy algorithm is adopted, the privacy issue cannot be resolved because the original data are stored entirely on each node. Also, in replication algorithm, there is not redundancy since $P_{retrival}^{\min}$ is satisfied by only cloud storage.

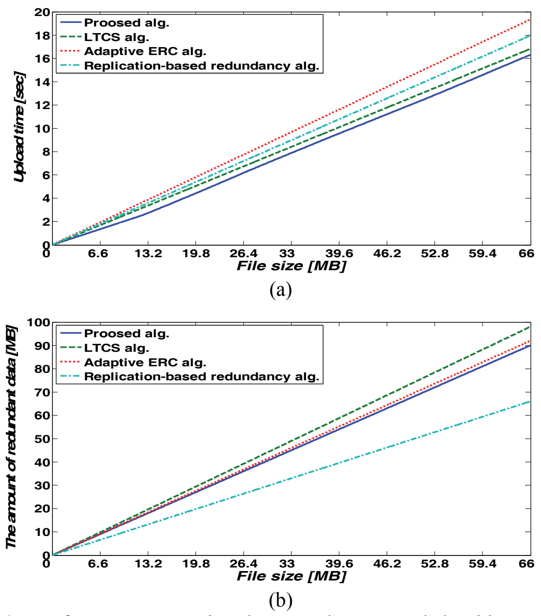

Fig. 2. Performance comparison between the proposed algorithm and other existing algorithms under the environment in Table I when $P_{retrival}^{\min}$ is 99.9999%: (a) upload time and (b) the amount of redundant data.

IV. CONCLUSION

In this work, we have introduced the hybrid P2P-cloud storage system. We employ a fountain code as a component to prevent the invasion of privacy, and take advantage of the characteristics of rateless codes to improve data retrieval. It has been observed during the simulation that the proposed algorithm enables fast completion of the upload transmission while satisfying the required data retrieval and supporting the privacy of user data.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2013R1A1A2006732) and the MSIP(Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2014.

REFERENCES

[1] J. Li and Q. Huang, "Erasure resilient codes in peer-to-peer storage cloud," IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 4, 2006.

[2] N. Cao, S. Yu, Z. Yang, W. Lou and Y. T. Hou, "LT code-based secure and reliable cloud storage service," IEEE INFOCOM , pp. 693–701 , 2012.

[3] M. Luby, "LT codes," Annual Symposium on Foundation of Computer Science, pp. 271–280, 2002.

[4] N. Ntene and J. H. V. Vuuren, "A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem," Discrete Optimization, vol. 6, pp. 174–188, 2009.

[5] F. E. Bustamante and Y. Qiao, "Friendships that last: peer lifespan and its role in P2P protocols," International Workshop on Web Content Caching and Distribution, pp. 233–246, 2004.

[6] D. J. C. Mackay, "Fountain codes," IEE Proceedings–Communications., vol. 152, pp. 1062–1068, 2005.

[7] L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud Storage as the Infrastructure of Cloud Computing," Intelligent Computing and Cognitive Informatics, pp. 380–383, 2010.