Storage and Availability Aware Fragment Placement for P2P Storage Systems

Kevin Sijo Puthusseri, Adhrit Shetty, Mrunmayee Patil and Kailas K. Devadkar

***Abstract*—Cloud storage systems are of great importance today due to features such as availability, mobility and performance. In addition to this, peer-to-peer systems ensure fault tolerance and scalability. The ideal combination of them is a P2P cloud storage system with an autonomous, self-configuring and fault-tolerant network. Besides performance, incentives are a key aspect of such P2P storage systems, with money being the most commonly used one. This paper puts forward the idea of a shared interest in data as the only incentive in order to build a viable P2P storage solution for both personal and enterprise storage. However participation in such a system is voluntary, making replication critical to ensuring data availability in the face of node churn. Erasure coding is a famous way of increasing data durability while maintaining a low expansion factor. The distribution of these encoded fragments though, across multiple peers is an opportunity for optimization which dictates much of the system performance. The paper proposes a fragment placement strategy to optimize the usage of peer resources and ensure symmetry in the system. For the purpose of node lookup and resource discovery in the network, an implementation of Kademlia distributed hash table is used.**

***Index Terms*—Peer-to-Peer, Kademlia, Erasure Coding**

I. Introduction

STORAGE solutions these days are bound by the capacity of a local machine(s) or the money invested in cloud storage. Making it an expensive process for both users of personal computers and large-scale enterprises. Also the client-server paradigm offers limited resilience in contrast to what a peer-to-peer model can. Not to forget, data security is defined by the ability of the finite points of failure, the servers, to handle breaches.While P2P storage systems have been studied for a long time, the practicality of these systems have always been a point of conflict [1]. However the growing need for a more decentralized and robust system, coupled with the significant increase in the number and capacity of personal computers today, has made it possible to conceive a viable P2P storage solution. Not only do the collective resources of several personal computers beat the capacity of any a single server or group of servers, the distributed nature of data storage adds an additional layer of security through anonymity [2].

Such storage solutions usually have a monetary incentive scheme for the participant peers who provide storage space. However attaching money to storage reverts back to the same model utilized by cloud storage providers. The probability of obtaining low cost storage will keep decreasing as the size of demanded storage and number of users increase. Instead leveraging the overlap of content desired by several individuals to provide a storage solution for data in which the users have a shared interest, would be a better model. Due to the lack of monetary incentives, the system can safely assume a much higher churn rate, which makes data availability more of a challenge. The answer to this is increasing the data redundancy in the system. But the approach taken to provide redundancy cannot be unaware of factors like per peer local storage and bandwidth consumption, as optimizing the usage of peer resources is the only way to build a practical solution. Another important aspect of P2P systems is symmetry [3]. Being able to carry out storage and network operations without burdening any particular peer unfairly is extremely important. Both symmetry and resource optimization are primarily defined by the strategy of data placement in the network.

With this in consideration, the paper proposes a storage solution that builds on existing P2P technologies and practices, while satisfying the additional demands of a shared interest incentive scheme. To achieve this goal we introduce a new fragment placement algorithm that tackles the above mentioned problems. For node lookup and resource discovery in the P2P network we have used an implementation of Kademlia which is a type of distributed hash table. It routes queries and locates nodes using a XOR based metric topology that has displayed shorter lookup path lengths in multiple applications [4]. It also provides resistance to denial of service attacks due to its consideration of node alive time while filling the peer's routing table [5]. These characteristics make it an ideal selection as the backbone of our P2P network.

The rest of this paper is structured as follows. Section II contains related work on reliability in P2P storage systems. Section III covers the operations fundamental to a P2P storage solution. Section IV defines the proposed solution. Section V is the pseudocode of the fragment placement algorithm. Section VI discusses the performance of the solution. Section VII states the conclusion of the paper.

Kevin Sijo Puthusseri, Mrunmayee Patil, Kailas K. Devadkar and Adhrit Shetty are with Department of Information Technology, Bharatiya Vidya Bhavans Sardar Patel Institute of Technology, Mumbai, India. (e-mail: kevinsijo@gmail.com adhritshetty2908@gmail.com mrunz21@gmail.com and kailas_devadkar@spit.ac.in)

978-1-5386-7595-3/19/$31.00 ©2019 IEEE

II. Related Work

Data churn is the fast oscillations in P2P systems caused by the rapid joining and leaving of nodes. Churn leads to decreased data availiability, subsequent increase in access latency and eventual loss of data, reducing overall system performance [3]. Redundancy mechanisms serve as a solution to the fore-mentioned problem. Introducing redundancy and distributing it among peers in the network is a critical operation of P2P systems [6]. This can be achieved by using either replication or erasure codes [7]. Erasure coding is more effective when compared to replication because it provides higher data durability at lower expansion factors [8], [6]. For this very reason we chose to use erasure coding for this solution. Once the file is replicated or erasure coded the fragments have to be stored across multiple peers. The placement of these fragments is extremely important as it decides the access latency, the probability of data availability and the cost of data reconstruction.

Numerous replica placement algorithms have been proposed and studied over the years. Strategies like [9], [10] enhance data availability, but considers only user online characteristics. In [11] the replica placement is based on patterns of data availability of the peers over a certain period of time. While it does improve data availability in some cases, it fails for situations where there is no significant pattern in peer uptime. Moreover strategies for placing replicas do not generate the same results when applied to erasure coded fragments. This is due to the difference in number of fragments required for data access and reconstruction. The objective of this paper is to perform erasure coded fragment placement that ensures reliability while considering peer symmetry in terms of storage, bandwidth and compute consumption.

III. P2P Storage Operations

A. File Encryption and Hashing

Data in the network should only be visible to the participants and not any external rogue entity. This entity could either be sniffing on the network to intercept data in transit or try to duplicate one of the legitimate peers. To enforce data security and prevent such situations from occurring encrypting the data is a minimum. The most commonly used symmetric key algorithm in P2P systems is Advanced Encryption Standard. All valid peers have the symmetric key using which they can both upload and download files to and from the network. In case a peer doesn't have the key, he will not be allowed to upload a file and will not be able to view the data.

To ensure data integrity, in addition to encrypting it the system hashes the content and pushes these hashes onto the network. These hashes can then be used to ensure that the data is not corrupted, as they serve as checks of integrity. When a peer requests for a file, the hash stored on the network is compared with the hash of the content downloaded on his device. In case it matches the data is uncorrupted, otherwise the same data has to be obtained from another source. Any secure hashing algorithm (SHA) can be used for this purpose.

B. File Fragmentation

File content stored on the peer nodes is vulnerable to brute force attacks. In spite of encrypting the file, the data is not completely secure. For this purpose the file data after encryption is fragmented and stored on the individual peers. This adds an additional layer of security as the fragments by themselves are of no use. At the same time dividing the file into smaller fragments provides the system much more granularity with respect to load balancing the storage operations between the network peers. Besides fragmentation, redundancy creation in the system is also necessary. For this purpose we make use of Reed-Solomon Erasure Coding. The K,N ratio is decided based on the durability expected from the system. In case the file size is so big that the K,N ratio leads to greater than desired fragment sizes, the file will have to be segmented into more moderate sizes, which can then be put together to obtain the original file. The K,N ratio along with other fragment and file metadata is pushed onto the distributed hash table and made available to the nodes in the network in accordance with Kademlia's protocol for P2P communication and resource lookup. The process of erasure coding a file or segment can be memory and compute intensive. So to ease the load, the file or segment first undergoes striping. Striping involves dividing the file into pieces of fixed size, applying erasure coding on the individual pieces and then combining them to form N total fragments. At the time of reconstruction we will require any K of these fragments which will be decoded to form the original file or segment.

C. Data repair

While erasure coding does ensure a certain level of durability in the face of node failure, the system needs to be prepared for data loss. This basically is loss of fragments due to peers leaving the network permanently. To maintain a certain number of fragments for each data item at all times, we can have either eager or lazy data repair strategies. In an eager strategy we start the data repair upon losing a single fragment. While in lazy repair we decide a repair threshold and perform data repair only when the number of fragments fall below that. It has been found that lazy repair with a suitable threshold offer the same level of reliability at reduced bandwidth utilization [6]. Data repair requires K uncorrupted fragments to be downloaded to a single peer and the original file to be reconstructed. Once this is done then the missing or corrupted fragments can be obtained by erasure coding the reconstructed file.

D. Peer Lookup and Resource Discovery

Kademlia makes use of a 160-bit ID to identify the different peers and resources in the network. At each peer is a routing table that contains the nodes IDs and other details needed to locate other peers. The routing table consists of multiple k-buckets each covering a certain range of the node ID space. The metric considered while selecting these ranges is the XOR distance between the other node ID and its own. This XOR distance signifies closeness in Kademlia. Each k-bucket contains $k_{closest}$ most recently alive nodes at any time. The

value of $k_{closest}$ generally used is arbitrarily chosen to be 20. Whenever Kademlia performs a lookup for peers either to distribute or retrieve data, it considers the $k_{closest}$ nodes to its own ID. We utilize this policy with slight alterations while distributing data in our system.

IV. Storage and Availability Aware Fragment Placement

The important mathematical terms and symbols are specified in Table I.

TABLE I
MATHEMATICAL MODEL OF THE ALGORITHM

Symbols	Meaning
C	Cycles of Simulation
P	Number of peers
S	Size of file
K	Number of required fragments (Erasure coding)
N	Total number of fragments (Erasure coding)
M	Repair threshold
α	Data expansion factor
p_i	Peer ($i \epsilon [1, P]$)
a_i	Probability of p_i being online
s_i	Normalised value of storage consumed of p_i
$k_{closest}$	Numbers of peers selected due to XOR closeness to an ID
d_i	Normalised value of XOR distance of p_i from fragment
Θ_i	Peer selection metric for p_i
g_j	Group ($j \epsilon (1, k^N_{closest})$)
r_i	Number of times p_i repeats itself in g_j
δ_j	Repair cost metric for g_j
λ_j	Group selection metric for g_j

A. Problem Definition

We consider a system that consists of P peers which is simulated over C cycles. Let a_i be the probability of a peer p_i (i [1, P]) being online, S be the size of the file uploaded during each cycle and (K, M, N) represent the essential Erasure coding constants. N represents the total number of fragments formed as a result of Erasure Coding the file, K represents the minimum number of fragments required to rebuild the file and M is the threshold of available fragments below which data repair is undertaken. Kademlia forms the backbone for all routing and discovery related operations and uses a 160-bit keyspace to assign IDs to peers and resources. Closeness between a peer x and resource (fragment) y is defined by the XOR distance between their IDs, i.e, xŁy. In contrast to Kademlia's arbitrary choice of the bucket size containing $k_{closest}$ peers, we select $k_{closest}$ as N^α where α = Expansion Factor = N/K. Varying $k_{closest}$ enables controlling the search space in order to optimize for not only closeness (XOR distance) but also other parameters. Each time a file is to be uploaded, it is encrypted using AES, fragmented using Erasure Coding and distributed using any suitable method (which, for Kademlia is, finding the closest peer). For each of the N fragments, we have a bucket of $k_{closest}$ peers. Now, since our goal is to achieve system-wide symmetry for all parameters (storage, availability and bandwidth), we need to choose one peer from each of these N buckets such that as a group they contribute to the overall symmetry of the system without sacrificing general availability of uploaded file. We investigate a group fairness-based scheme to select the group of N peers, from the possible $k^N_{closest}$ groups, that has the high closest overall file availability, minimal XOR distance and burdens peers with lower storage utilization while incurring minimum data repair cost associated with failure of $N - M$ peers.

B. Peer Evaluation Metric

The optimality of choosing a peer to host a fragment is dependent of how available it is, how much of its storage is utilized and how far is it from said fragment.

In order to evaluate a peer's optimality a combined metric Θ_i is used as specified in Equation 1.

$$\Theta_i = (1 - a_i) + s_i + d_i \tag{1}$$

Θ_i represents how unsuitable a peer is as a host, with equal importance given to availability, storage and distance. Thus, choosing k-closest peers based on Θ_i becomes a minimization problem. In order to ensure equal importance of each parameter, storage utilized and distance from fragment are normalized to values within $[0, 1]$ just as peer availability. Peer availability governs the probability of successfully accessing the fragment hosted by it. Higher the availability, higher is the probability of retrieving the fragment. To construct Θ_i, we consider peer unavailability, $1 - a_i$, since it contributes to the unsuitability of the peer. Overall storage utilization symmetry is achieved by penalizing any peer choice having already high storage utilization thereby making it less likely to be the chosen candidate. s_i is normalized with respect to the maximum peer storage utilization amongst the $k_{closest}$ peers. Since distance is intrinsic to Kademlia's functioning and in order to minimize number of hops required to access a given fragment from the closest peer, d_i contributes to Θ_i as the XOR distance normalized with respect to the maximum value in the key space (2^{160}).

C. Group Selection Metric

The optimality of choosing a group of N peers to host N fragments from N buckets of $k_{closest}$ peers is a multi-objective based optimization making it a Pareto Frontier problem. To compare the $k^N_{closest}$ possible groups, we use a Simple Culling algorithm generally used to solve non complex Pareto Frontier problems. The primary concern of using Erasure Coded fragments is that once the number of accessible fragments of a file fall below the repair threshold M, data repair has to be carried out. Data repair is both compute and bandwidth expensive. Hence the lesser it is required the better it is for peer resource consumption. In order to involve this critical aspect in the evaluation process, we calculate a metric, δ_j, signifying the worst repair cost for each group g_j. It is the probability of the most unreliable $N - M$ fragments of the group becoming unavailable. Since multiple fragments may be hosted by the same peer, it is necessary to penalize such a

choice given that it hampers overall file retrievability. Using distinct sets of $N-M$ peers, we calculate δ_j for each one and penalize each distinct peer's unavailability with respect to the number of fragments the selection proposes it must host. The value of this metric is calculated using equation 2. The set with the highest calculated value is considered the most unreliable and its metric value is chosen as δ_j for the group. Lower the value of δ_j, better is the suitability of the group for fragment placement.

$$\delta_j = \prod_{i=1}^{N-M} (1-a_i)^{(1/r_i)} \tag{2}$$

Overall optimality of a group depends on its worst repair cost and average peer metrics of its members. Thus, to evaluate the optimality of the group g_j as a whole we define a group selection metric λ_j calculated using equation 3. The group with the lowest value of λ_j is the optimal selection for placement of the N fragments.

$$\lambda_j = \delta_j + (\sum_{i=1}^{N} \Theta_i)/N \tag{3}$$

V. Pseudocode

The pseudo code of the placement of fragments is shown in Algorithm 1.

Algorithm 1 Fragment Placement

Input: fragment list
Initialization: $candidatelist$;
while $i \leq N$ **do**
  $candidates_i = kClosestPeers(fragment_i)$;
  while $j \leq k_{closest}$ **do**
    FInd normalised storage utilization for $candidate_j$;
    Find normalised XOR distance for $candidate_j$;
    Find θ for $candidate_j$ using formula (1);
  end while
end while
Find potential groups for storing N fragments
while $i \leq k^N_{closest}$ **do**
  Calculate peer count in $group_i$;
  Find combinations of $N-M$ fragments in $group_i$;
  while $j \leq numberofcombinations$ **do**
    Calculate δ for $combination_j$ using formula (2)
    if δ<δ_{worst} **then**
      $\delta_{worst} = \delta$
    end if
  end while
  Calculate λ for $group_i$ using formula (3)
end while
Select group with lowest λ as optimal selection

VI. Performance Study

We have simulated the fragment placement strategies of proposed in this paper and Kademlia for over 2000 cycles, adding one file per cycle to each system to show the performance gains by using the proposed strategy. Both systems start with 100 peers each where each peer has an online availability, a_i, within $[0.2, 0.9]$ and starts with nil storage occupied. We make use of K=4, M=6 and N=8 as global erasure coding constants. Thus, our value of $k_{closest} = 64$ since α is $N/K = 2$.

On observing the system characteristics through the simulation, we focus mainly on Storage Symmetry and File Retrievability. Storage Symmetry refers to the difference of storage values between the highest mean storage peer and lowest mean storage peer through all cycles, i.e, the area between the lines representing the increase in occupied storage of both the highest mean storage peer and lowest mean storage. File Retrievability refers to the best possibility of retrieving a file at any given time, i.e, the best combined online probability of K peers given N peers.

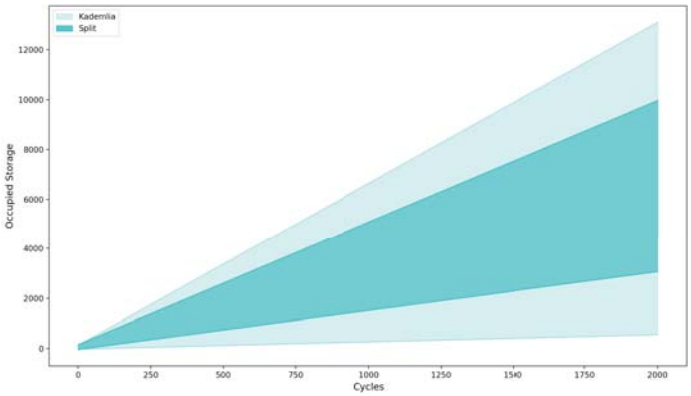

Fig. 1. Storage Symmetry Comparison

Fig. 1 shows that occupied storage trends of Kademlia are in more stark contrast when compared the those of the proposed system, thus exemplifying how the proposed system enforces Storage Symmetry.

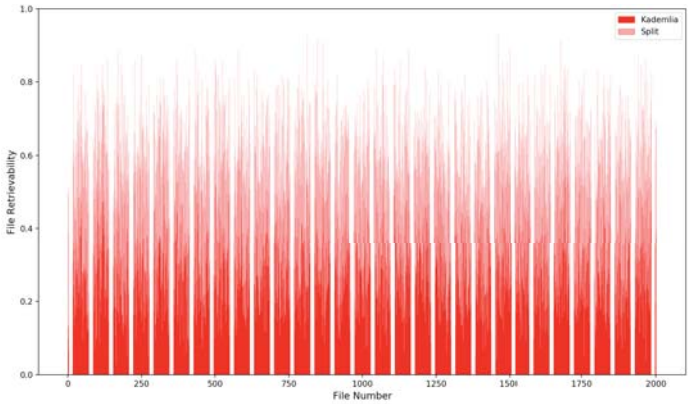

Fig. 2. File Retrievability Comparison

Fig. 2 shows that file retrievability for each added file is approximately 30% more in the our system as compared to Kademlia, thus proving that availability is not compromised entirely in the efforts of achieving system wide symmetry.

VII. Conclusion

The paper proposes a storage and availability aware fragment placement strategy for P2P storage systems. The proposed system, achieves better symmetry for parameters such as storage and distance while increasing overall file retrievability when compared to a Kademlia-based P2P system. Nevertheless, the inherent Pareto Frontier problem of an optimal group selection can be solved significantly more efficiently by making of use of state-of-the-art heuristic alternatives. Since a real-world P2P system involves higher number of increasingly dynamic parameters which may impact performance directly such as Bandwidth capacity, Geographic positioning, Online Availability Patterns, future improvements certainly include consideration of these parameters as well.

References

[1] Vincze, G., Pap, Z., & Horvath, R., "Peer-to-peer based distributed file systems," *International Journal of Internet Protocol Technology,* vol.2, no.2, pp.117, 2007.

[2] P. Boch and J. afak, "Security and reliability of distributed file systems," *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems, Prague,* 2011, pp. 764-769. Heidelberg

[3] A Survey of Peer-to-Peer Storage Techniques for Distributed File Systems. [Online]. Available: http://courses.cs.vt.edu/cs5204/fall08-kafura/Papers/FileSystems/Survey.pdf. [Accessed 6 March 2019]

[4] E. Harjula, T. Koskela and M. Ylianttila, "Comparing the performance and efficiency of two popular DHTs in interpersonal communication," *2011 IEEE Wireless Communications and Networking Conference, Cancun, Quintana Roo,* 2011, pp. 2173-2178.

[5] P. Maymounkov and D. Mazires, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," *Peer-to-Peer Systems Lecture Notes in Computer Science*, vol. 2429, pp. 5365, Oct. 2002.

[6] F. Giroire, J. Monteiro, and S. Perennes, "Peer-to-Peer Storage Systems: A Practical Guideline to be Lazy," *2010 IEEE Global Telecommunications Conference GLOBECOM 2010,* 2010.

[7] H. Weatherspoon and J. D. Kubiatowicz, Erasure Coding Vs. Replication: A Quantitative Comparison," *Peer-to-Peer Systems Lecture Notes in Computer Science,* pp. 328337, 2002.

[8] Rizvi, S. S., & Razaque, A, "Black-box: A new secure distributed peer-to-peer file storage and backup system," in *19th International Multi-Topic Conference,* 2016.

[9] K. Rzadca, A. Datta, and S. Buchegger, Replica Placement in P2P Storage: Complexity and Game Theoretic Analyses," *2010 IEEE 30th International Conference on Distributed Computing Systems,* 2010.

[10] S. Bernard and F. L. Fessant, Optimizing peer-to-peer backup using lifetime estimations," *Proceedings of the 2009 EDBT/ICDT Workshops on - EDBT/ICDT 09,* 2009.

[11] G. Song, S. Kim, and D. Seo, Replica Placement Algorithm for Highly Available Peer-to-Peer Storage Systems," *2009 First International Conference on Advances in P2P Systems,* 2009.