Replica Placement in Peer-Assisted Clouds: An Economic Approach

Ahmed Ali-Eldin, Sameh El-Ansary

Replica Placement in Peer-Assisted Clouds: an Economic Approach

Ahmed Ali-Eldin and Sameh El-Ansary

Center of Informatics science, Nile University
ahmed.alieldin@nileu.edu.eg,sansary@nileuniversity.edu.eg

Abstract. We introduce NileStore, a replica placement algorithm based on an economical model for use in Peer-assisted cloud storage. The algorithm uses storage and bandwidth resources of peers to offload the cloud provider's resources. We formulate the placement problem as a linear task assignment problem where the aim is to minimize time needed for file replicas to reach a certain desired threshold. Using simulation, We reduce the probability of a file being served from the provider's servers by more than 97.5% under realistic network conditions.

Keywords: Peer to peer computing; cloud storage

1 Introduction

Cloud storage systems are online storage systems where data is stored on groups of virtual servers rather than dedicated servers. The storage provider assigns resources for a user according to the **current** requirements of the customer. Cost of bandwidth is the strongest challenge facing cloud storage [1]. One of the answers to this challenge is to build a peer-assisted cloud where Peers' resources are used to offload the storage servers. The provider distributes replicas of the data through the network to reduce the cost of operation, provide fault tolerance and provide a reliable service at reduced costs.

Providing guarantees on the availability and durability in a system depending on volatile peers is hard. Availability is the ability of a peer in the network to retrieve a data object at any time. Durability of a data object represents the time the object is not permanently lost. A P2P storage system gives probabilistic guarantees on the durability of 97% of the data [2]. In a peer-assisted approach, data will always be stored on the servers of the storage service provider so there will also be guarantees on availability. In this paper we focus on the problem of replica placement in peer-assisted cloud storage. Replica placement addresses the problem of where to place the replicas created by the system to maintain highest levels of durability and availability of the data stored. The main contributions in this work are: 1- We introduce an economical formulation for the replica placement problem.2- We introduce Nilestore, a Peer-assisted cloud storage network protocol that offloads the service provider's servers. We show that using Nilestore can results in at least 75% improvement over using a random placement algorithm.

2 Related work

In [3], a peer-assisted cloud storage system deployed in china is introduced. The authors describe the system design and show some measurements. Our work can be considered as an extension to their system as FS2you uses random placement for the replicas. Toka *et al.*[1], prove that using this peer-assisted clouds can provide a performance comparable to that of a centralized cloud at a fraction of the cost. For placement, they cluster peers depending on their online behavior. Our system takes into account the contributed storage, bandwidth and the scarcity of the data when doing data placement and aims at offloading the servers of the service provider.

Oceanstore [2] and Farsite [4] are examples of P2P storage systems. In [5], the authors prove that when redundancy, data scale, and dynamics are all high, the needed cross-system bandwidth is unreasonable. A similar conclusion was presented in [4]. These results makes peer-assisted cloud storage systems a more attractive approach as the storage nodes in the cloud are less dynamic compared to the P2P nodes.

3 Replica placement and Economics

The problem of replica placement in the context of peer-assisted cloud storage can be defined as follows: Given a group of peers in a cloud storage network where each peer have some data for replication, free storage and unused bandwidth. Make r replicas of the data of each peer using the contributed space of the other peers in a way that increase the amount of data retrieved from the peers compared to that retrieved from the servers.

The economic problem is the problem involving the allocation of scarce resources among different alternatives or competing ends [6]. Replica placement is similar to the economic problem as there are scarce resources (bandwidth and storage) that can be allocated between the different peers-different alternatives. Replica management economies are systems where the peer acquires real or virtual money for hosting replicas of others. The machines will use this money to buy storage space for its own replicas [7]. We consider the problem of replica placement in peer-assisted storage clouds as an economical problem. We try to solve it using a mixture of two types of auctioning; first-price sealed-bid auctioning and double auctioning [8]. In first-price sealed-bid auction, each buyer submits one bid for a resource with no knowledge of the bids of the other buyers. The highest bidder wins and gets the resource for the price he specified in his bid. In a double auction, bidders submits their bids while sellers submit the items (resources) each will sell at anytime during the trading period. If at any time any of the bids is matching with any of the sold items (quantity of sold resources), the trade is executed immediately.

In our Nilestore, we consider three main players; the cloud provider as the auctioneer and peers who play a dual role; sellers who contribute resources to get money and use it for buying backup as bidders. All peers send a sealed bid to the

cloud provider containing the amount of contributed resources and the amount of needed resources. If fairness is to be imposed, a peer is not allowed to send a bid in which the amount of resources he contributes is less than the amount of resources he buys. A peer can place a bid where his contributed resources are more than the amount of resources he plans to buy in order to be sure that he has a higher bid than the others. The provider receiving this bid during a trading period will match the different bidders with the different sellers in a way that maximizes the utility.

4 Players design

In Nilestore, the allocation server holds an auction every τ time units. The peers send sealed bids specifying data blocks to be uploaded including any data blocks hosted, amount of storage to be contributed to the system, peer's upload bandwidth, peer's download bandwidth, a list of hashes of the data objects and the amount of contributed resources. A data block is replicated r times to provide higher availability. The list of hashes identify the replica count available from each data object. Deduplication is achieved using the replica count of each data object. After τ time units, the server does not accept more bids for the current trading period. Any late bids are stored for the evaluation during the next round of bids.
The server converts the replica placement problem to a task assignment problem [9]. We calculate the profit of allocating the resources of a seller peer to every available buyer.

4.1 The profit function

The profit function between a buyer peer p_i and a seller peer p_j consists of three multiplied terms. The three terms are:

1. **The Feasibility of storage S_{ij}:** is responsible for capturing the feasibility of storing the data of p_i of size $|\hat{B}_i|$ on the free space F_j on p_j where $\hat{B}_i$ is the list of blocks that the buyer wants to replicate and $|\hat{B}_i|$ is its size. When coupling two peers based on the storage, we want to reduce the fragmentation thus we try to keep the blocks owned by a peer spatially on the same machines. This allows a peer to contact a minimal number of peers to retrieve all the data he owned. The system should look for best fit allocation between the peers. In a system where fairness is important, A peer contributes at least r times the size of data he initially wants to replicate. If the peer chooses to increase his storage contribution, he will host more blocks and eventually get a higher utility. The feasibility of storage is thus calculated as follows:

$$S_{ij} = \frac{\min(|\hat{B}_i|, F_j)}{\max(|\hat{B}_i|, F_j)} \tag{1}$$

2. **The Feasibility of transfer T_{ij}:** This term adds the bandwidth consideration to the utility calculations. We to couple peers such that their bandwidths is maximally utilized to reduce the transfer time . Fairness is imposed by enforcing a ratio between the upload bandwidth of p_i and the download bandwidth of a peer. The term is:

$$T_{ij} = \frac{\min(u_i, d_j)}{\max(u_i, d_j)} \tag{2}$$

3. **The average scarcity of the blocks of a buyer $H_{un}(p_i)$:** This term represents the scarcity of the data block of a peer p_i. we define the scarcity of a peer to be the average number of replications that its blocks need. That is,

$$H_{un}(p_i) = \frac{\sum_{\forall b_{ik} \in \hat{B}_i} r - R(b_{ik})}{|\hat{B}_i|} \tag{3}$$

where $\hat{B}_i$ is the set of blocks owned by p_i that needs replication, b_{ik} is a single block on p_i and $R(b_{ik})$ is the number of replicas available for block b_{ik} in the network.

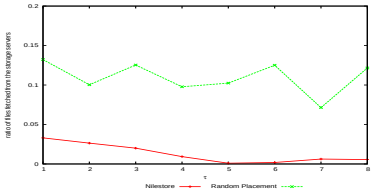
These three terms are multiplied to form the local profit and are fed into an assignment engine which tries to find a suboptimal allocation policy that maximizes the profit for the system while satisfying the needs of every buyer. The goal is to reduce consumption of the bandwidth of all different peers while reducing the load on the storage servers. We started experimenting using the Hungarian algorithm [9] which proved not suitable because of its complexity $O(n^4)$. We designed a greedy suboptimal task assignment engine that has lower complexity of $O(n \log n)$ where n is the number of peers.

4.2 Solving the task assignment problem

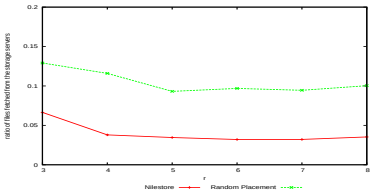
The algorithm shuffles the list of workers randomly. The first worker on the top of the list is picked and all of the jobs that he can perform are sorted according to decreasing profit. The job with the highest profit is assigned to the selected worker and the job's name is added to a list containing the names of all the assigned jobs. Second worker on the list is then picked, the jobs he can do are sorted and is assigned to the non-assigned job with highest profit. This is repeated for every worker in the workers list such that no job is assigned to two workers.

5 Simulation and results

We built a discrete event simulator that simulates peer-assisted cloud storage networks. We used a group of P2P workload studies to come to a workload model. Peer bandwidths were obtained from [10]. We generate for each peer a number of unique objects. The contributed free storage for a peer is set randomly



(a) Effect of changing the trading period on the performance on Nilestore versus random placement



(b) The effect of increasing the number of replicas on performance

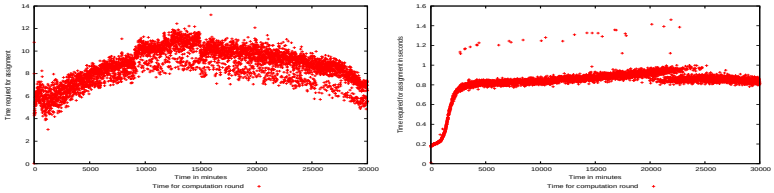
Fig. 1. Nilestore performance

between r and $2r$ times the size of his data . We used [11]and [12] to quantify the peer Join/Leave rates.

We conducted experiments to evaluate Nilestore versus random placement. To the best of our knowledge, random placement is the only approach for replica placement used in the peer-assisted storage literature. In random placement, a peer replicates his data on a peer chosen randomly from the peers available in the network. Figure 1(a) shows the ratio of data blocks unavailable after 20000 bidding rounds with trading periods varying between one minute and eight minutes. The figure shows that using Nilestore improves the system performance by 70 to 90%. If τ is small, many of the peers will not be able to send their bids and the assignment algorithm will have less options. If τ is chosen to be very large, Nilestore will react to failures slowly risking the loss of the objects that need replication. It can be seen from the figure that choosing $\tau = 5$ reduces contact with the servers by almost 95%. Figure 1(b) shows the effect of increasing the threshold r for the number of replicas made for each data block in the system. Our simulation results conform with the previous results [12] on the number of replicas needed. Figure 2 shows that 8 seconds are needed for making the allocation when there are 1000 peers in the system.

6 Conclusion and Future work

In this work, we introduced Nilestore, a Peer-assisted cloud storage protocol that offloads the resources of the cloud storage servers using the unused resources of the peers subscribed to the storage service. A single copy of each data object is stored on the storage servers to ensure durability and availability and duplicates are distributed across the network. Peers always try to retrieve data from the network before contacting the servers. In the future we plan to distribute the



(a) Time required for the replica placement computation using Nilestore for 1000 peers

(b) Time required for the replica placement computation using random placement for 1000 peers

Fig. 2. The time of allocation in Nilestore and using a random placement approach

allocation process and to deploy the system in real-life and consider the trust levels of the peers.

References

1. L. Toka, M. Dell'Amico, and P. Michiardi, "Online Data Backup: A Peer-Assisted Approach," in *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*. IEEE, 2010, pp. 1–10.
2. J. Kubiatowicz, "Extracting guarantees from chaos," *Communications of the ACM*, vol. 46, no. 2, pp. 33–38, 2003.
3. Y. Sun, F. Liu, B. Li, B. Li, and X. Zhang, "Fs2you: Peer-assisted semi-persistent online storage at a large scale," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 873–881.
4. W. Bolosky, J. Douceur, and J. Howell, "The farsite project: a retrospective," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 2, pp. 17–26, 2007.
5. C. Blake and R. Rodrigues, "High availability, scalable storage, dynamic peer networks: Pick two," in *Proceedings of the 9th conference on Hot Topics in Operating Systems-Volume 9*. USENIX Association, 2003, p. 1.
6. J. Buchanan, "What should economists do?" *Southern Economic Journal*, vol. 30, no. 3, pp. 213–222, 1964.
7. D. Geels and J. Kubiatowicz, "Replica management should be a game," in *In Proc. of the 10th European SIGOPS Workshop*. ACM, 2002.
8. R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," *Concurrency and computation: practice and experience*, vol. 14, no. 13-15, pp. 1507–1542, 2002.
9. R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2009.
10. D. K. Correa, "Assessing Broadband in America: OECD and ITIF Broadband Rankings," *SSRN eLibrary*, 2007.
11. M. Steiner, T. En-Najjary, and E. W. Biersack, "Analyzing peer behavior in kad," Institut Eurecom, October 2007, Tech. Rep.
12. B.-G. Chun, F. Dabek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris, "Efficient replica maintenance for distributed storage systems," in *NSDI'06: Proceedings of the 3rd conference on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2006, pp. 4–4.