I. J. Computer Network and Information Security, **2013, 5,55-63**
Published Online April 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2013.05.07

Recent Replica Placement Algorithms in P2P Networks – A Review

Kala Karun. A, Dr. Jayasudha. J. S
Sree Chitra Thirunal College of Engineering, Thiruvananthapuram, Kerala, India
kalavipin@gmail.com, jayasudhajs@yahoo.com

Abstract — Peer-to-Peer (P2P) systems provide a platform for large internet scale distributed applications with no dedicated infrastructure. In P2P systems, applications are developed by dividing them across individual systems termed peers which take the role of both client as well as server. Popularity of objects in such a system can change rapidly, which demands the need for a rapid and light weight content replication strategy which considers this dynamic popularity changes. While considering P2P in distributed file sharing applications, data availability has significant impact on the system's performance. In addition to optimized availability, the replica placement should guarantee reduced search and data access latency. It should be dynamically adaptable to instantaneous query arrival rate as well as dynamic membership of individual peers. Also, it should provide good performance with reduced number of control messages. Thus an intelligent placement of replicas considering various factors of the system always outperforms a random placement of replicas on random peers. This paper reviews and compares various recent replica placement algorithms for structured as well as unstructured P2P networks.

Index Terms — Replication, Replica Placement, Content Distribution Network, Peer to Peer

I. Introduction

P2P systems represent a network of peers connected by communication channel, with each peer takes the role of the client as well as server of a client-server model. P2P systems have proved their efficiency in large scale decentralized file sharing applications over the internet. They are able to adapt the arrival and departure of nodes with relatively low cost and are robust and self organizing.

The attractive features of these systems are there high availability as well as reduced query latency towards user requests. These are achieved because of the inherent redundancy in the system through replication where peers replicate each other's data so that when one peer is offline the other can serve the request. Many studies in P2P networks consider the replica placement problem i.e. how to place replicas in proper locations so that the overall performance of the system is increased. The proposed algorithms consider various aspects of the like available capacity of peers, popularity of elements to be replicated, bandwidth of peers, availability of peers etc. while taking the replica placement decision.

The goal of replica placement algorithms (RPA) also varies from each other. The goal may be to maximize the availability of peers, to make popular objects highly available or improve the QoS of the system. Such an intelligent RPA achieves much more performance than random RPA's. There are static as well as dynamic RPAs [1]. Static RPA replicates objects statically into certain peers which can be accessed by other peers. An RPA that considers the dynamic aspects of the system seems to be more efficient. Replication strategies can be centralized or distributed. In centralized replication, the replica placement decisions will be taken by a centralized node whereas in distributed replication, all the nodes in the system participate in taking the decision.

To make P2P highly efficient, a slight variation in its pure architecture is applied and yields P2P-CDN architecture. This hybrid architecture combines the complimentary benefits of P2P as well as Content delivery networks (CDN). They are also called Hybrid CDN-P2P architecture, or HCDN. Pure CDN networks replicate popular objects by deploying surrogate servers at the edge of the internet, which has the disadvantages of increased deployment cost. On the other hand pure P2P suffers from degraded QoS guarantees at the worst case. Researchers are also looking for improved RPA for P2P-CDN architecture for making it highly efficient. Such RPA's are also considered in this review.

The paper is organized as follows. Section II deals with different replica placement algorithms in P2P networks. Section III gives the comparison of the data replication techniques reviewed in section II. Several factors like goal, architecture, system model, design methodology, time complexity etc. are compared and given in Table II. The paper concludes in section IV.

II. Replica Placement Algorithms

A. Replica Placement Based on Peer Availability Table (RPAT)

Building a highly available P2P system, especially a P2P storage system is difficult as well as challenging as the peers can join and leave the system at any time without any notice. To meet this challenge a replica placement algorithm which exploits the availability pattern of each individual peer is considered in this algorithm [2]. The availability pattern of individual peers has been traced through a probabilistic model referred to as Peer Availability Table (PAT), which simply covers short term as well as long term availability of peers under consideration. The replication algorithm estimates the similarity between PAT's and based on a heuristic approach, and places replicas at proper peers so as to make the system highly available.

Availability of needed data is a critical requirement in any peer-to-peer application. However, ensuring high availability in P2P system is quite difficult. At a particular moment of time assume that several peers are not available. Then, generally the data availability would be decreased. In this scenario, the data availability can be improved if the remaining available peers take the responsibilities for storing and providing the required data as a substitute for unavailable peers. This idea is used in this replica placement algorithm, which place replicas at proper peers by exploiting the availability pattern of individual peers so that the system remains available even though when several peers are down. This algorithm selects peers, which have the most different but reasonable availability pattern to store the replicas. A probabilistic model called PAT (Peer Availability Table) is used to represent a peer's availability.

Peer Availability Table (PAT) is a probabilistic model which represents peer availability. PAT can reflect the diverse aspects of peer characteristics and can cover short-term availability to long-term availability in a simple manner. It can be used to find peers who have similar usage patterns or to detect the permanent departure of a peer. As there is no central server to evaluate all peers' availability in the P2P system, each peer manages a personal PAT and neighbors' PATs only when they are online. The process of PAT management is as follows. A peer's PAT is initialized to zero when it is bootstrapping to the P2P storage system for the first time. At that time, the peer gets a routing table that contains information about neighbor nodes. As per the table, the peer creates PATs for neighboring nodes. The peer's PAT is actively updated every 5 minutes by itself, while the neighbring nodes' PATs are updated passively. The passive method is based on a heartbeat message. All nodes in the systems send a heartbeat message (or keep-alive message) to their neighbor nodes to inform them that a node is available periodically. Peers who receive a heartbeat message from another node increase the verification counts and the online counts of the sender's PAT at the corresponding time slot to the received time. Following these processes, all nodes' PATs are maintained up-to-date. A sample PAT table is shown in Table I.

Table I. Example of pat

Time Slots	0	1	..	1000
Verification Counts	55	22		30
Online Counts	18	15		25

The availability of a particular peer with time slot i is calculated by

$$AV_i = \text{Online Counts} / \text{Verification Counts} \quad (1)$$

Peers are very dynamic (i.e. Each peer has a diverse PAT) and can be classified by the degree of similarity of PATs. Because there are a large number of peers in the P2P system, it is possible that there exist some peers who have a similar PAT. The degree of similarity between peer A and peer B is calculated by Similarity (A,B) = $PA_i * PB_i$, where PA_i and PB_i are the availabilities of peer A and peer B at time slot i. A high degree of similarity between peer A and peer B means that their usage patterns are very much alike. The replica placement algorithm begins at this point. Selecting peers with a high degree of similarity ensures high availability when trying to get data at a usual usage time for the owner. While selecting nodes, these nodes' average availabilities should be more than the threshold to ensure minimum availability since the similarity will be zero if calculated with a peer whose availability converges to zero. Therefore, if the degrees of similarity were less than a threshold, that combination would be discarded.

The algorithm uses PAT for the replica placement. Given the file Id and the target availability requirement of the file, the algorithm outputs the list of peers which maximize the availability of the file. The procedure Find the candidate-list of all nodes which can keep a copy of the file and calculates the similarity between each node in the candidate-list and discard those pairs having similarity less than a predetermined threshold value. Then the similarity-list is sorted and peers having a similar availability pattern is grouped. A representative peer from each group is selected and the replica is placed until the targeted availability is met.

Advantages

- ➢ The algorithm finds the best combination of peers to provide the highest data availability among candidate peer.
- ➢ Compared to a random placement scheme, the proposed algorithm improves data availability with moderate overhead in terms of memory consumption and processing time.
- ➢ Permanent departure of peers can be detected with the help of PAT.

Limitations

- ➢ Errors in the network are not considered and assumed that data transmission is completed at once.
- ➢ The size of candidate list and target availability is directly given as parameter.

B. Online Pointer Replication (OPR) Algorithm in P2P Networks (OPRA)

This algorithm [3] effectively reduces the worst case query latency using online pointer replication. The degree of replication achieved by OPR is dynamically adjusted to the instantaneous query arrival rate and characteristics of the system so as to reduce total control traffic. In an environment where the popularity of objects changes dynamically, OPR provides a rapid and light weight content replication strategy to reduce search and data access latencies. Based on a greedy approximation algorithm, OPRA intelligently replicate roots where pointers to objects are distributed.

A large distributed network which has various objects like video files, web pages, documents etc. is considered. Object queries can be originated at any node at any time and if the querying node is aware of the object location, the query is considered successful. To facilitate the object search, a location-pointer of the object is stored in a peer called root in the network. Thus, all queries for that object will be forwarded to the root node to get the location pointer. Latency incurred by a query is denoted by the number of hops it takes to reach the root. To reduce the query search latency and to improve data availability, the location pointers kept in the root are replicated.

When the number of replicas increases, the search latency decreases, but the memory consumption as well as the communication cost for pointer updates increases. The replication overhead includes memory overhead as well as communication overhead. Memory overhead is mainly associated with the storage of location pointer and the communication overhead is mainly in keeping the pointers up-to-date which is more significant. In this algorithm, a replication strategy that minimizes the communication cost is being proposed. The algorithm calculates the shortest distances between the original root r_1 and all the other vertices in graph G and then peers to replicate the root is identified.

It is proved that the optimal replication degree is directly proportional to the query arrival rate and inversely proportional to the system churn rate. As a result, depending upon the dynamic environment changes, the degree of replication can also be tuned dynamically. The query arrival rate and the system churn rate can be determined using the sampling technique proposed in [4].

Advantages

- Low latency while locating resources and less overhead of control messages.
- Effective storage utilization.
- Highly scalable solution and can tune the replication degree.

Limitations

- Considers only peer locations and no other dynamic aspects of the system or popularity of objects.
- Centralized and workload constraints are not considered.
- Mobility of nodes are not considered. The nodes may leave due to link disconnections.
- High network bandwidth consumption.

C. Replica Placement Algorithm for Hybrid CDN-P2P Architecture (HCPA)

The algorithm [5] is proposed for the Hybrid CDN-P2P architecture, or HCDN. The algorithm considers the effects of P2P distribution at the P2P level and deployed surrogate servers at CDN level. Compared to replica placement algorithms for pure CDN networks, the proposed approach can reduce the placement cost (i.e. The deployment cost) since the peer contribution is taken into account.

The hybrid structure shown in Fig. 1 can be described as a graph G, which consists of the set of nodes V and the set of edges E. The set V can be divided into two sets, the set of surrogate servers V_s (with cardinality $|V_s| = N$) and the set of peers V_p. Let O = {F_1, F_2... F_M} denote the set of files which can be downloaded and shared by peers. The set S_i (with cardinality $|S_i| = n_i$) represents the optimal set of surrogate servers storing file F_i where as P_i represents the set of peers downloading file F_i. The goal of the proposed algorithm is to find the optimal set of surrogate servers S_i for each file F_i by minimizing the total placing cost for content transmission and storage. The placement cost, (PlaceCostv, i) for file F_i in node v depends on the transport cost and storage cost. Transport cost depends on the distances to other nodes storing file F_i and storage cost depends on the size of the file which is to be stored. It is assumed that the replica placement algorithm is executed at time T with perfect knowledge of the traffic pattern during the time interval [0, T].

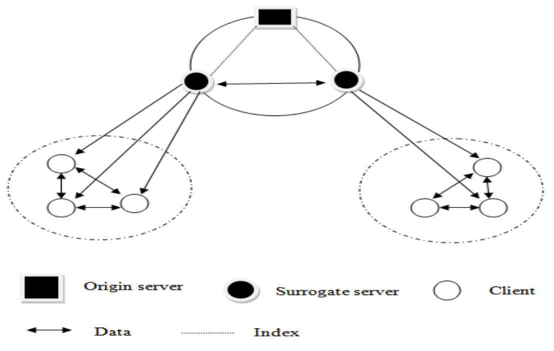

Figure 1. Network model of HCDN

Peers are classified into two categories: downloaders and seeds. Downloaders are peers having parts of a file and seeds are peers having entire file. Seeds will reside in the network to allow other peers to download from them. The factor η ($0 \leq \eta \leq 1$) is used to represent the average uploading effectiveness of all down loaders. $\eta = 1$ means that the downloader's uploading rate is equal to their available outgoing bandwidth and $\eta = 0$ means that the down loaders do not upload data to other peers at all.

The proposed solution considers a homogenous environment in which each peer having the same uploading bandwidth is denoted as 'u'. So, in the P2P-level network, the total uploading rate of all peers for file F_i is $(x_i + \eta y_i)$ u, where x_i and y_i are number of seeds and down loaders for file F_i during [0, T] respectively. Now the number of user requests m_i that can be handled by the current peer contribution is given by:

$$m_i = (x_i + \eta y_i) u / d_i, \qquad (2)$$

Where d_i is the data rate of file F_i.

The placement cost can be computed as:

$$\text{PlaceCost}_i = \begin{cases} b_i(r_i-m_i)d_s(S_i)+\alpha s_i n_i, & (r_i>m_i) \\ 0, & (r_i>m_i) \end{cases} \quad (3)$$

Here, $d_s(S_i)$ denotes the average distance between the nearest surrogate server and the surrogate servers storing F_i. The cost function has two parts, the transport cost $b_i(r_i-m_i)d_s(S_i)$ and storage cost $\alpha s_i n_i$. The total distribution cost also involves the transport among the peers.

$$\text{TotalCost}_i = \begin{cases} b_i(r_i-m_i)d_s(S_i)+ b_i m_i d_p(P_i)+\alpha s_i n_i, & (r_i>m_i) \\ b_i r_i d_p(P_i), & (r_{i<=}m_i) \end{cases} \quad (4)$$

Here, $d_p(P_i)$ denotes the average distance between the requesting and supplying peers. If $m_i > r_i$, the total cost will be $b_i r_i d_p(Pi)$ which is equal to the transport cost among the peers. $m_i=0$ means that the peers do not contribute to the uploading service. From equations 2 and 3,

$$\text{TotalCost}_i = \text{PlaceCost}_i= b_i(r_i)d_s(S_i) + \alpha s_i n_i \quad (5)$$

For each file F_i, calculate m_i and r_i. If ($r_i <= m_i$) then no need to replicate the file anywhere as the current peer contribution can satisfy all the expected user requests. Otherwise, evaluate ($\text{PlaceCost}_{v,i}$) for each surrogate server v for file F_i and then find the optimal set of surrogate servers for placing the replicas so that the total placement cost is minimized. The objects can be now placed on the identified set of optimal surrogate servers.

Advantages

- Minimizes replica placement cost.
- By considering peer contribution, the system reduces deployment cost in pure CDN as well improves QoS in pure P2P.
- Increases availability, lowest backbone traffic, reduces congestion in backbone links.
- Low access latency and communication overhead.

Limitations

- It is not in agreement with the characteristic of the internet as an open medium for public information issuance.
- Construction and maintenance cost is high.
- Single point of failure.
- The central server can easily get overloaded.

D. Clustered K- Center Replica Placement Algorithm (CKCA)

The algorithm [6] considers the replica placement in real world P2P networks as a clustered K center problem which is NP complete. It is an efficient static and distributed approximation algorithm which can work with several orders of magnitude faster than optimal solutions with minimum access latency. The k center problem tries to find a set of k centers in an arbitrary graph such that the maximum shortest distance of all nodes to the nearest center is minimized.

The system consists of a set of n peers in a distributed P2P network and it is assumed that queries can originate in any node and are forwarded to the nearest node where the needed file is present using some existing mechanism [7]. The objective of the algorithm is, given the replica count k , find the k nodes R= $r_1,r_2,\ldots,r_k$ such that the maximum access latency is minimized, i.e. the aim is to make the network a forest of k search trees with each tree rooted at one replication node r_i. This algorithm creates a partition of m non-overlapping groups $g_1, \ldots, g_m$ of k centers where m= n/k, n is the total number of nodes in the P2P network. The algorithm first computes the distance graph G_d = (V, E_d) which is a completely connected graph where the weight of any edge v_i-v_j is the sum of the total edges` weight in the shortest path from v_i to v_j in the given network computed using Dijkstra's algorithm. Now sort all the edges in G_d in ascending order. The next step is to find a subgraph of Gd with clusters of k centers using an iterative procedure.

For every iteration i

1. Add an edge from the list of sorted edges to form the graph G_i and remove the added edge from the sorted list.
2. Check whether the graph G_i contains an appropriate feasible sub graph for clusters of k centers. The test is to check for k connected components each with size m and can be done using DFS.
3. If such a sub graph is not present go to step 1 else exit.

Place the replicas in the identified k centers. If the P2P is having huge network traffic, then the selected k centers will easily get overloaded. So it is not a good idea to use the same k centers for all the objects. In that case, execute the same algorithm after excluding the selected k center to form a new set of k centers.

Advantages

- Improves the performance of P2P system by minimizing the maximum query latency across the network.
- Considers the capacities of K centers and place replicas accordingly.

Limitations

- Highly sensitive to network topology.
- The dynamic nature of the system is not considered.
- The storage capacity of the k centers gets saturated as replication increase.

E. A K Coordinated Decentralized Replica Placement Algorithm (KCDA)

This algorithm [8] was proposed for a hybrid P2P-CDN network or HCDN. The proposed solution algorithm aims at introducing an efficient replica placement for HCDN that will minimize the overall deployment cost of surrogate servers and increases the net QoS guarantees. The algorithm tries to obtain maximum gain by placing replicas suitably for a network with assumed ring topology. Each surrogate server makes the replica placement in terms of the content replicas on k closer surrogate servers, which will improve the system scalability compared to the centralized replica placement heuristics.

The HCDN considered in the algorithm follows a hybrid topology where the surrogate servers at CDN level form a ring topology and the origin server, CDN level surrogate servers and P2P level peers respectively form a hierarchical alignment.

The network is represented as a graph G (V, E), with set of vertices V and set of edges E. Vertices are divided into two sets: set of surrogate servers V_s and set of peers V_p. There are M files F = {F_1, F_2 … F_m}, which are subscribed to the servers and downloaded by the peers. The algorithm aims at obtaining the optimal set of files F_v stored on a specific surrogate server v. Each surrogate server gathers workload information from the local P2P network during the time interval [0, T]. r_i^v denotes the total number of user requests for file i collected by surrogate server v during [0, T]. Peers are divided into two sets: leechers and seeds. Leechers are peers who are downloading content from other supplying peers and servers, while seeds have already obtained the whole file and are still providing the upload service for requesting peers. x_i^v and y_i^v denote the number of seeds and leechers for file i connected to surrogate server v during [0, T]. The factor ŋ (0≤ ŋ≤1) represents the average upload capability of peers. If ŋ equals one, it means that peers can use all their available outgoing bandwidth to serve other peers and if ŋ equals zero, it means that peers cannot serve other requesting peers. The size of file 'i' is 'S_i' and its data rate is 'b_i' which is able to satisfy the quality of service of peers.

The network environment is assumed to be homogenous where each peer has the same upload bandwidth u. Gain (v, i) denotes the gain when the file i is replicated on the surrogate server v. Now the number of user requests served by peers connected to surrogate server v, denoted as m_i^v is given by:

$$m_i^v = (x_i^v + ŋ\, y_i^v) / b_i \qquad (6)$$

The paper proposes a k-coordinated decentralized replica placement algorithm (KCDA) based on the gain formulation of the replica placement problem.

$$\text{Gain}(v,i)=\begin{cases} b_i(r_i^v- m_i^v)\, d_s(\text{closest}(v),i)-d_s(v,i)-\alpha s_i, & (r_i^v> m_i^v) \\ -\alpha s_i, & (r_i^v> m_i^v) \end{cases} \qquad (7)$$

Here, d_s(closest(v),i) represents the average distance between the requesting peer and the surrogate server which is closest to the surrogate server v and storing the file i. d_s(v,i) represents the average distance between the requesting peer and the surrogate server v. (r_i^v - m_i^v) represents the user requests which need to be served by the servers. A represents the trade-off between the transport gain and storage cost and αs_i denotes the storage cost. Factor k denotes how many surrogate servers will be associated with the CDN-P2P architecture when one surrogate server makes the replica placement. Periodically, each surrogate server gathers the local workload information such as user request patterns and network link delay. Then, it calculates the total upload capability of peers and the gain. User requests not satisfied by supplying peers in the P2P network should be served by closer surrogate servers at the ring-based CDN level. If there are no replicas of a specific file stored on closer surrogate servers, user requests can be served by the origin servers. The replica placement is decentralized so each surrogate server needs to gather only the local workload information and interact with k closer surrogate servers. The k-coordinated heuristic is executed by each surrogate server periodically which replicates the files having more gain to the local surrogate server.

For each server and for every file f_i , m_i^v and r_i^v is determined and then the deployment gain of files at server v is calculated. The file with larger gain will be replicated on the surrogate server v and send the replica distribution information to the closer k coordinated surrogate servers which collect the workload information during the time interval [0, T]. The Mandelbrot-Zipf distribution [9, 10] is used to find file popularity on the P2P network.

Advantages

- ➢ Maximizes gain obtained out of replication and increases availability.
- ➢ By considering peer contribution, the system reduces deployment cost in pure CDN as well improves QoS in pure P2P, lowest backbone traffic, reduces congestion in backbone links.
- ➢ Low access latency and communication overhead.

Limitations

- ➢ Replication decisions are taken mainly based on popularity only.
- ➢ Dynamic nature of nodes and network is not considered.

F. A Cost Effective File Replication in P2P Systems (CEFR)

In P2P file sharing systems, file sharing occurs without the interference of the central server. Increased number of hot spots demands file replication to reduce access delays. Hot spots are popular files which are requested frequently by the clients. The algorithm [11] presents an effective file replication that increases query efficiency and replica utilization at low cost. This is achieved through the selection of hot spots, frequent requesters as replica nodes and dynamically adapting to non uniform and time varying file popularity.

The algorithm replicates files in the overloaded areas and there will be an adaptive adjustment of replica nodes when there are time varying file popularities. It tries to enhance the replica hit rate, look up efficiency and reduces the overhead. A file will be replicated based on the query rate of the file and a threshold value which is a constant parameter. The algorithm adaptively adjusts the replica nodes based on the recent query traffic in a decentralized manner to deal with varying popularity. The query rate of a file is denoted by q_f which is the number of requests generated per unit time. A threshold value T_q is set for the query rate which is given by $T_q=\alpha * avg_q$ where α is a constant parameter, α >2 and avg_q is the average query rate in the system given by

$$\mathrm{avg}_q = \sum_{j=1}^{n} (qf_j) \quad n \; / \qquad (8)$$

Server nodes can act as both file owner and replica nodes. The visit rate of a node indicates the number of requests the node has received during time period T. If the request rate of a file is greater than the threshold value i.e. if $q_f > T_q$ then the node is considered as a frequent requester of the file.

For efficient replication, a decentralized approach is used. The creation and removal of a replica can be done by calculating the query rate of a node. If the number of requests in a node is less, the underutilized replica will be removed. If the request rate of a node $qf > \delta T_q$ where δ is the under loaded factor i.e. $\delta < 1$, the replica will be marked if this condition occurs once. If this condition continuously occurs for a particular time period, it will be removed.

A file requester will periodically calculate file request rates and if the request rate of a file is greater than the threshold and then if that file is queried, include a replication request into the query. The query forwarding node also will periodically calculate file request rates and if request rate is greater than the threshold and if a query for forwarding that file is received, then include a replication request into the query. Each file server will periodically calculate load l_i and if it is overloaded by a factor γ and if no file replication requests are present during time T, replicate file to the neighboring nodes that frequently forward requests for the file. If it is overloaded by a factor γ and if file replication requests are present during T order the requesters in descending order of request rates and replicate file to the requester on top of the list and remove it from the list. Each replica node will calculate the request rate for each replica of file periodically and if it is less than the threshold, it will be removed.

Advantages

- Query efficiency is increased.
- Reduces underutilized replicas.
- High replica hit rate than client side algorithms.

Limitations

- Placement decision is taken mainly based on request rates only.
- Careful selection of the threshold value is required.

G. Band Width Aware Replica Placement for P2P Systems (BWAR)

The paper [12] proposes a new approach for replica placement where the available bandwidth of individual peers, online characteristics of individual users as well as the popularity of the elements to be replicated is considered during replica placement. The solution achieves increased data availability as well as improved data access probability for individual data items.

The replication system assumed by the paper is group reciprocity based similar to [13] where the users are arranged into different groups and each user in a particular group replicates data items of all other members of the group. In such a group the availability of a particular data item can be calculated to be the probability that at least one member in the group is online. The proposed algorithm divides the users into different groups by considering the popularity of the data they replicate as well as the available bandwidth they have so as to get a mutual balance between them. The major goals of the replication strategy are to classify the users into different reciprocity based groups so that data availability of the worst group can be maximized and to improve the group such that when the group is available, the available members should have enough bandwidth to serve the entire request for the data it replicates.

To achieve the second goal the system introduces a new term called access probability to compute the probability that at least one online candidate still has enough bandwidth to serve a requested data. For each data item stored in a particular group, the access probability depends on the online members of the group which possess the data. The aggregated bandwidth B of the online members determines the access probability of the data, i.e. the number of requests for the data that can be supported by the group. A Larger value of B indicates high access probability of the data. Now find the aggregated data access rate A of all data replicated in the group. If A is greater than B, it is very likely that a request to the replication group will be dropped due to the limited bandwidth of online members. Similarly the access probability of each and every data item can be evaluated by considering all possible subsets of the given group.

To solve the problem with reasonable complexity a swap-based heuristic (SBH) is suggested. The heuristic classifies users into different replication groups such that all data items have a similar high access probability. To design such a system, initially, the users are assigned to different replication groups randomly. Then, the system starts swapping procedure that swaps two members from different groups repetitively. To select the swapping groups two reciprocal groups with the highest and lowest access probability respectively are chosen. Now to choose which pair of users to be swapped, the procedure exhaustedly searches every possible pair of users from the reciprocal groups and swap them, then it compute the new access probability for the modified reciprocal groups. Now the user pair that can minimize the absolute difference between newly calculated access probabilities of the reciprocal group will be selected for the swapping heuristic.

The above procedure requires the information about access probability for all groups and data items, and thus might cost a high message overhead if the system scales up. To reduce this overhead, the above swapping procedure can be loosened to a distributed version, called the distributed swap-based heuristic (D-SBH), which apply the same approach to select users to be swapped, while choosing two swapping groups randomly without any global knowledge. Due to the lack of global information, the distributed algorithm D-SBH might require more swapping iterations to achieve a similar performance as compared to SBH.

To further balance the trade-off between performance and message overhead, a slight variation of the above technique is considered, called hybrid SBH (H-SBH). In H-SBH, it randomly sample H replication groups in every swapping iteration, and select the group with the highest access probability and the group with lowest access probability from H sampled groups to perform swapping. In other words, if H is set to the total number of groups in the system, then the H-SBH and the original SBH scheme are alike. On the contrary, if H is set to 2, H-SBH is then equivalent to D-SBH. Intuitively, a larger H requires a higher overhead to collect information about groups, while achieving a better performance with fewer swapping iterations.

Advantages

- By jointly considering bandwidth of individual peers, the online characteristics of individual users as well as the popularity of the elements to be replicated, the algorithm has significantly higher performance.
- It guarantees increased availability as well as improved data access probability.

Limitations

- Dynamic membership not considered.
- Storage capacity constraints are not considered.

III. Comparison of replica placement Algorithms

Comparison of the data replication techniques reviewed in the previous sections is given in Table I. Several factors like goal, architecture, system model, design methodology, time complexity etc. are compared to the different replica placement algorithms like RPAT, OPRA, HCPA, CKCA, KCDA, CEFR & BWAR.

IV. Conclusions and future work

The paper gives an overview of replication and replica placement problems in the emerging P2P distributed computing environment. Replica Placement Problem aims to place replicas strategically yielding maximum benefits from replication. The focus of this paper is to give an overview of various centralized/distributed and static/dynamic replica placement algorithms in structured as well as unstructured P2Ps. The paper compared 7 popular P2P replica placement algorithms and extracted their main features. Each of the replica placement strategies considered in this paper aims at different goals and optimize various aspects of the system, making P2P highly efficient. The paper finally extracted and summarized major features behind each RPA.

The current RPA's can be extended by adding mechanisms for propagating updates efficiently across replicated copies of the updated objects. I.e. required consistency rules can be proposed depending on the application. Either strict or eventual consistency models can be selected. The maintenance of replication transparency is another issue, i.e. to make the system behave as if there is no replication within the system; i.e. the complexity of replication should be hidden from users.

References

[1] M. Tang, B. Lee, C. Yeo, X. Tang, "Dynamic replication algorithms for the multi-tier Data Grid", Future Generation Computer Systems - Special issue: Parallel computing technologies , 21, 5, May 2005, pp. 775-790.

[2] G. Song, S. Kim, D. Seo, "Replica Placement Algorithm for Highly Available Peer-to-Peer Storage Systems," Proceedings of First International Conference on Advances in P2P Systems, AP2PS '09, 2009, pp. 160 -167.

[3] J. Zhou, L. N. Bhuyan, A. Banerjee, "An effective pointer replication algorithm in P2P networks," IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, pp. 1 – 11.

[4] B. Arai, G. Das, D. Gunopulos and V. Kalogeraki, "Approximating Aggregation Queries in Peer-to-Peer Networks," Proceedings of the 22nd International Conference on Data Engineering (ICDE'06), 2006.

[5] H. Jiang, Z. Wang, A. K. Wong, A.K, J. Li, Z. Li. "A Replica Placement Algorithm for Hybrid CDN-P2P Architecture," 15th International Conference on Parallel and Distributed Systems, 2009, pp. 758 – 763.

[6] J. Zhou, X. Zhang, L. N. Bhuyan, B. Liu, "Clustered K-Center: Effective Replica Placement in Peer-to-Peer Systems", Proceedings of the Global Communications Conference, GLOBECOM '07, Washington, DC, USA, November 2007, pp. 26-30.

[7] X. Tang and J. Xu, "On replica placement for QoS-aware content distribution", Proceedings of Twenty-Third Annual Joint Conference of the IEEE Computer and Communication Societies, 2004, pp. 806-815.

[8] Z. Wang, H. Jiang, Y. Su, J. Li, J. Liu, Dutkiewicz, Eryk, "A k-coordinated decentralized replica placement algorithm for the ring-based CDN-P2P architecture", Proceedings of IEEE Symposium on Computers and Communications (ISCC), 2010, pp. 811 -816.

[9] K. Gummadi, R. Dunn, S. Saroiu, et al. "Measurement, modeling, and analysis of a peer-to-peer file-sharing workload," Proceedings of 19th ACM Symp. Operating Systems Principles (SOSP'03), Oct. 2003.

[10] M. Hefeeda, and O. Saleh. "Traffic modeling and proportional partial caching for Peer-to-Peer systems," IEEE/ACM Transactions on networking, Vol. 16, No. 6, Dec. 2008.

[11] P.G. Jeyasheeli, L. Rajashree, "Cost Effective File Replication in P2P File Sharing Systems", Proceedings of International Conference on Computing, Electronics and Electrical Technologies, March 2012, pp. 948 -952.

[12] T. Y. Chih, K. C. Lin, C. C. Fu, "Bandwidth-Aware Replica Placement for Peer-to-Peer Storage Systems",

Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 2011), 2011, pp. 1-5.

[13] K. Rzadca, A. Datta, and S. Buchegger, "Replica placement in p2p storage: Complexity and game theoretic analyses," in IEEE ICDCS, 2010.

Kala Karun. A. She is pursuing M. Tech. in Computer Science and Engineering from Sree Chithra Thirunal College of Engineering under University of Kerala, India. Her research interests include Distributed Computing and Algorithms.

Dr. Jayasudha J. S. She is the Head of the Department and Professor in Department of Computer Science & Engineering, Sree Chithra Thirunal College of Engineering Under University of Kerala, India. She received her Ph. D Degree from University of Kerala. Her research interests include Computer Networks, Neural Networks etc.

Table II. Comparison of Replica Placement Algorithms

Features	RPAT	OPRA	HCPA	CKCA	KCDA	CEFR	BWAR
Goal	Ensure high availability.	Reduce query latency and control traffic.	Minimize total placement cost	Minimize access latency	Reduce back bone traffic and enhance scalability	Increase query efficiency and replica utilization at low cost.	Improve data avialbility and access probability
Architecture	Centralized	Centralized	Decentralize d	Centralize d	Decentralized	Decentralized	Centralized / Decentralized
Data Object	Objects with any granularity	Pointers to stored objects	Files	Files	Files	Files	Objects with any granularity
Read only Assumptions	Yes	Yes	Yes	Yes	Yes	Update is also considered	yes
Static/Dynami c	Dynamic	Static	Dynamic	Static	Dynamic	Dynamic	Dynamic
Parameters considered	Availability of peers, Availability requirement of objects	Communicati on cost, system churn rate, query arrival rate, pointer updating cost.	Transport cost, storage cost, distance to nodes	Replicatio n factor k, distance between the nodes	Workload, no. of coordinated surrogate servers, no. of user requests for a file, distance between peers and surrogate servers.	Request rates, threshold	Online characteristics, data popularity, bandwidth capability
Popoularity/ Threshold Based	-	-	Popularity	-	Popularity	Threshold	Popularity
System Model	Structured / unstructured P2P with random topology	Unstructured P2P with random topology	Hybrid CDN-P2P architecture with hybrid topology	P2P system with graph topology	Hybrid CDN-P2P architecture with hybrid topology (ring + hierarchical)	Structured / unstructured P2P with random topology	Any P2P System
Design Methodology	Greedy	Greedy	Greedy	Greedy	Greedy	Greedy	Greedy
Number of Replicas	Computed by the algorithm	Computed by the algorithm	Computed by the algorithm	Given as input	Given as input	Computed by the algorithm	Computed by the algorithm
Quality of Service	-	-	-	-	QoS is improved	-	-
Workload Balancing	-	-	-	-	Considered	-	Considered
Capacity Constraints	-	Limited storage capacity	Limited storage capacity	Limited storage capacity	Limited storage capacity	Limited storage capacity	limited capacity

Features	RPAT	OPRA	HCPA	CKCA	KCDA	CEFR	BWAR
Time Complexity	Proportional to peer availability and searching PAT.	O(K(N+E) log N)where n- no. peers selected for storing pointers, N- total no. of peers, E – total no. of links connecting the peers	O(mn) where m- no. of files to be replicated & n- no. of surrogate servers	O(n^4) n- no. of nodes in the network	O(mn) where m- no. of files to be replicated & n- no. of surrogate servers	O(mn), m- no. of files, n- no. of servers	O(s^2), s – max. storage capacity of any peer.