Malugo: A peer-to-peer storage system

Yu-Wei Chan, Tsung-Hsuan Ho, Po-Chi Shih and Yeh-Ching Chung*

Department of Computer Science,
National Tsing Hua University,
Hsinchu 30013, Taiwan, ROC
E-mail: ywchan@sslab.cs.nthu.edu.tw
E-mail: ansons@gmail.com
E-mail: shedoh@sslab.cs.nthu.edu.tw
E-mail: ychung@cs.nthu.edu.tw
*Corresponding author

Abstract: We consider the problem of routing locality in peer-to-peer storage systems where peers store and exchange data among themselves. With the global information, peers will take the data locality into consideration when they implement their replication mechanisms to keep a number of file replicas all over the systems. In this paper, we mainly propose a peer-to-peer storage system – Malugo. Algorithms for the implementation of the peers' locating and file operation processes are also presented. Simulation results show that the proposed system successfully constructs an efficient and stable peer-to-peer storage environment with considerations of data and routing locality among peers.

Keywords: distributed storage system; peer-to-peer storage; Malugo.

Reference to this paper should be made as follows: Chan, Y-W., Ho, T-H., Shih, P-C. and Chung, Y-C. (2010) 'Malugo: A peer-to-peer storage system', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 5, No. 4, pp.209–218.

Biographical notes: Yu-Wei Chan received his BS and MS in Information Engineering from TamKang University, Taiwan, in 1997 and 2001, respectively. He is currently pursuing his PhD in the Department of Computer Science, National Tsing Hua University, Taiwan. His current research interests include peer-to-peer computing, peer-to-peer streaming and game-theoretic resource allocation.

Tsung-Hsuan Ho received his BS from the Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan, in 2006, and MS from the Institute of Information Systems and Applications, National Tsing Hua University, Hsinchu, Taiwan, in 2008. He is currently working in the industry with internet multimedia development.

Po-Chi Shih received the BS and MS in Computer Science and Information Engineering from Tunghai University in 2003 and 2005, respectively. He is now studying PhD in Computer Science in National Tsing Hua University. His research interests include grid computing and peer-to-peer technology.

Yeh-Ching Chung received a BS in Information Engineering from Chung Yuan Christian University in 1983, and the MS and PhD in Computer and Information Science from Syracuse University in 1988 and 1992, respectively. He joined the Department of Information Engineering at Feng Chia University as an Associate Professor in 1992 and became a Full Professor in 1999. From 1998 to 2001, he was the Chairman of the Department. In 2002, he joined the Department of Computer Science at National Tsing Hua University as a Full Professor. His research interests include parallel and distributed processing, cluster systems, grid computing, multi-core tool chain design, and multi-core embedded systems. He is a Member of the IEEE computer society and ACM.

1 Introduction

With the exploding growth of internet, more and more global organisations share their resources and collaborate with each other in large-scale projects. One of the most fundamental challenges of these cross-organisation collaborations is that how to efficiently exchange data within physically distributed environment. One of the famous projects is SourceForge.net, which provides the downloading services for source programmes and documentations. Taiwan UniGrid (2006) system has been developed with a distributed storage platform to exchange data among several sites in Taiwan. However, a distributed storage system usually needs much cost and may consist of some bottlenecks such as scalability and reliability.

In the past few years, one of the most famous internet applications is the peer-to-peer technology. Peer-to-peer system is a kind of virtual overlay network built on application layer. Peers collaborate with each other for distributing data or working together by either structured or unstructured architecture. In recent years, significant peer-to-peer applications have been proposed, such as peer-to-peer file-sharing systems, peer-to-peer streaming systems and peer-to-peer storage systems.

Since the advantages of scalability and robustness of peer-to-peer system, a lot of peer-to-peer storage systems have been proposed such as Freenet (Clarke et al., 2000), OceanStore (Kubiatowicz et al., 2000), PAST (Druschel and Rowstron, 2001), CFS (Dabek et al., 2001) and CFR (Lin et al., 2007). To keep a number of replicas all over the system, some of these systems implement their replication mechanisms with the need of global information. In addition, to make the file popularity achieve different numbers of copies of different files. Some of them consider the data locality to achieve higher downloading rate while ignoring the routing locality issue.

In this paper, we mainly develop and propose an internet-based, reliable, scalable and efficient peer-to-peer storage architecture, which is called Malugo. This system consists of two layers. One is the bottom layer in which peers are clustered together to form groups and construct a ring topology with the Chord (Stoica et al., 2003) framework. The other is the upper layer in which groups of different regions are connected with each other by utilising a tree-like topology.

To cluster peers, we base on the architecture of mOverlay (Zhang et al., 2004) system to develop our overlay operations that when a new join peer wants to join a group, it will be able to choose a proper group or even create a new group automatically. In addition, new inserted files will be replicated to different groups and different files will have different numbers of replicas according to the pre-specified replication policy. Additional copies will be cached in different peers to balance the load of storage peers, which host popular files.

The contribution of this paper is as follows:

- We mainly proposed a peer-to-peer storage system, which addressed the routing locality issues, which have been overcome by our proposed replication mechanisms
- In this system, we have proposed efficient file cache and replication mechanisms to solve the hotspot problem and achieve reliable, scalable file distribution among peers.

The remainder of this paper is organised as follows. In Section 2, we compare some various systems with our system. In Section 3, we briefly introduce our system architecture.

Section 4 gives some system analyses with respect to our proposed system model. The simulation and experimental results are all shown in Section 5. Finally, some concluding remarks are presented.

2 Previous works

Many peer-to-peer data storage systems have been proposed in the past few years. Some peer-to-peer storage systems make use of the Distributed Hash Table (DHT) mechanism to collect nodes and organise them into an overlay. First, by using the consistent hash function such as SHA-1, each node or object will be given a key from hashing its address or object name. Then, they will map these keys to nodes that are belonged to them. When one peer requests an object, the DHT system calculates the key of this peer from requesting information (such as filename) and retrieve from corresponding node by some kind of efficient lookup operations. Besides these, it can balance the load of the system, since all nodes receive roughly the same number of keys.

Stoica et al. (2003) have designed a Chord system, which is one of the outstanding DHT systems and it is constructed by ring topology. The Chord system improves the scalability of consistent hashing by avoiding the need of maintaining fully connected neighbouring relationships. In the Chord system, the consistent hash function was used to collaborate with nodes and allocate objects to distribute data indices to achieve load balance among peers. In addition, the finger table mechanism accelerates routing speed of peers. These properties make the framework of Chord system suitable for constructing distributed data storage system.

Dabek et al. (2001) have proposed a CFS system, which is a Unix-style read-only file system layered on top of the Chord framework. Files stored in the CFS are split into several blocks to avoid the problem of storing a single large-scale file, which exceeds the capacity of nodes. Rowstron and Druschel (2001) have proposed a Pastry system, which is also based on the DHT scheme to route and deliver data. Druschel and Rowstron have presented a PAST system, which is a large-scale persistent storage system layered on the Pastry architecture. The PAST system can be layered on other routing architectures with some loss of locality and fault resilience properties.

Our system adopts the idea of the PAST system in the sense that our system uses the same method to store and replicate the whole file. But, our system is different from the PAST system. We do not rely on complex replication policies. Muthitacharoen et al. (2002) have designed an IVY system, which is a log-based file system that supports concurrent write operations. The IVY system uses the distributed hash scheme to store the logs efficiently.

Gupta et al. (2003) have proposed a Kelips system, which is a file system layered on its own routing scheme with O(1) lookup time. The fast lookup, however, suffers from the cost of larger memory usage and background communication overhead. Boundary Chord (Jin et al., 2005), HIERAS (Xu et al., 2003) system utilise the hierarchical ring architecture. The Boundary Chord system uses different layer names to partition different groups, and requests need to route among all layers. Both of them also construct a ring topology for upper overlay. The HIERAS system proposed a multiple-layered ring to reflect the network locality, but this system needs to maintain complex finger tables between layers.

Lin et al. (2007) have presented a CFR system, which is one of the peer-to-peer storage systems, which also bases on the Chord framework. Peers are all in the same ring topology in the CFR system, but they have different region ID numbers. When the data are inserted, they will be first placed on the corresponding peers whose ID has been assigned. Then, the data will be replicated to these corresponding peers. Our proposed system also adopts the idea of the CFR system. In the Malugo system, each region maintains a replica for each single file. However, the Malugo system is different from the CFR system. The difference is that administrators in the CFR system know geographical property and they can easily configure the locality settings. However, most administrators are failed to successfully and smoothly set up the storage servers from previous experiments. However, administrators in the Malugo system only need to set up a number of grouping criteria, server peers will automatically cluster together by themselves and help these system administrators for setting up their storage systems.

Furthermore, the main difference between our system and the above-described systems is the routing locality. This means that the above-described systems do not consider the routing locality of peers, but our system does. When we ignore the issue of routing locality, it will not only increase the lookup time but also introduce significant maintenance overhead, especially in the complex system such as the HEIRAS system. However, in the Malugo system, servers or client peers can automatically locate to a proper group so as to achieve high system performance.

3 System architecture

Our system consists of three modules, which are file management, intra-overlay and inter-overlay modules, respectively. The file management module is responsible for file operations such as file insertion, retrieve, recovery, replicate and cache from a storage peer. Details of this mechanism will be specified in the following Section 3.2. The inter-overlay module provides the ability for communication among groups. The intra-overlay module provides operations to locate peers to the proper groups using the scheme of the Chord system. Both modules will be described in Section 3.1.

3.1 Overlay construction

Our system is constructed in terms of two-layered architecture. The bottom layer called intra-group overlay is constructed with Chord framework. The overlay clusters neighbouring peers to provide services within local regions. The upper layer called inter-group overlay is constructed to connect local groups together with locality consideration. In our system, one of the peers in local group will be elected to be the root peer to handle interconnection between groups. Figure 1 shows an example of the overlay structure of our system. Table 1 lists the description of some terms in this paper.

Figure 1 An overview of system architecture

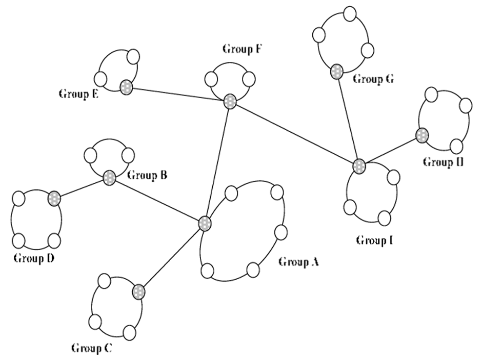


Table 1 Description of the terms in the system

Terms	*Description*
Access rate	Initial bandwidth between two peers (kbps)
Grouping bound	The pre-defined size of grouping bound when a new joining peer joins to the current group (kbps)
Contact peer	The root peer of a group which represents the location of the group

The grouping criterion of Malugo system is described as follows.

- When the distance between a new peer *Q* and group *A*'s neighbour is the same as the distance between group *A* and *A*'s neighbour groups, we say that the peer *Q* belongs to group *A*
- The distance could be represented as the bandwidth, response time, latency or some other terms to show the network status between any two peers.

In the following statements, we will introduce the peer-locating scheme of our system. Let us consider such a situation that when a new joining peer *N* wants to join a group, it will measure the access rate between it and the contact peer *C* of the group *G*. If the access rate between *C* and *N* is larger than *B*, we say that the peer *N* belongs to the group *G*. The algorithm of the peer-locating process is described as follows

A1 First, a new joining peer *N* will connect with the bootstrap peer of group *g*.

A2 If bootstrap peer is not the root peer of group *g*, it will redirect *N* to the root peer R_g. Otherwise, the bootstrap peer is the peer R_g.

A3 We measure the access rates between N-R_g and N-{R_g*'s* neighbours}, if the access rate between N-R_g is the largest one, we call the group *g* is the proper group for the peer *N*. Otherwise, we redirect the peer *N* to the peer R_j whose access rate is the largest one between it and the peer *N*.

A4 Repeat the above-described steps until the proper group has been found.

Although the locating process has finished, the access rate between peer *N* and the current proper group is still smaller than the group bound, which is represented by the character *B*. The new joining peer will notify the root peer of the selected group that there is a new group formed and it is a neighbour group. Other peers may connect to this new group to get data. Therefore, we also need to collect data from other neighbouring groups.

When a new group is constructed, one issue called topology mismatch may occur as illustrated in Figure 2(a). When a new peer *N* joins the overlay, it first connects with the peer *P1*. Then, it will connect with the peer *P2*. If *P2*'s group is too far away, the newly joined peer will form a new group by itself. Since *P2* is the last peer, which the newly joined peer connects, the newly joined peer will add the peer *P2* to its neighbour table.

From Figure 2, we can see that there are 2 links among *N*, *P1* and *P2*. If we can only keep 2 links among these 3 peers, the topology shown in Figure 2(b) is obviously better than Figure 2(a) since the total distance of {*P1-N*, *P2-N*} is smaller than {*P1-P2*, *P1-N*}. Liu et al. (2005) have proposed the LTM method to solve the above-described topology mismatch problem. We refer to the LTM method to handle the topology mismatch problem. The operations for coping with the topology mismatch problem are shown in Figure 3.

Figure 2 Topology mismatch problem: (a) an inefficient overlay and (b) a better connection model

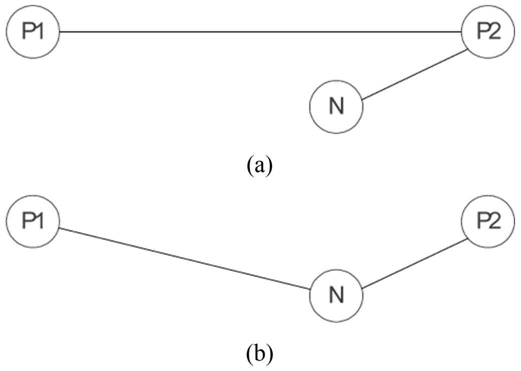


Figure 3 Pseudo-code for processing the topology mismatch problem

```
Procedure Fix_topology_mismatch()
{
  Speed = getSpeed(N, P2);
  neiP2 = P2.getNeighbor();
    For each peer in neiP2
    {
      If(getSpeed(peer, P2)<getSpeed(peer,
        N)) && speed>getSpeed(peer, P2)
        {
          delink(P2, peer);
          newLink(N, peer);
        }
      Else if (speed<getSpeed(peer, N) &&
        getSpeed(peer, P2)>speed))
        {
          delink(P2, N);
          newLink(N, peer);
        }
    }
}
```

If the access rate between the new joining peer *N* and the current proper group *G* is larger than the grouping bound *B*, *N* will join the local overlay and located itself by using the method proposed by Chord. Meanwhile, *N* will record the root peer information of group *G* for sending interactive information with other groups. The new joining peer *N* is now responsible for a section of keys, and therefore, we also need to retrieve data back from its successor peer.

Issues in typical super-peer architecture are single point of failure and hotspot if there is only one super-peer that is responsible for a group. In case of single point of failure means that once the root peer fails, the inter-group connection will be out of services since there is no more available gateway for relying messages. The hotspot problem represents the situation in which there are substantial connections, which connect to one specific peer suddenly. These connections will make use of this hotspot peer to search, relay or get files. In such a situation, the performance of this hotspot peer will degrade immediately. Therefore, we replicate the outgoing routing table to the predecessor and successor peers of this hotspot peer to keep the files availability and durability. The predecessor and successor peers will be served as the backhaul root peers of normal peers in the same group.

3.2 File operations

The algorithm of the file insertion process is described as follows.

B1 A file that was given a fileID will be inserted into the peer n_i whose id is the closest to the fileID within a specific group

B2 Peer n_i will notify its root peer n_r to start the replication process

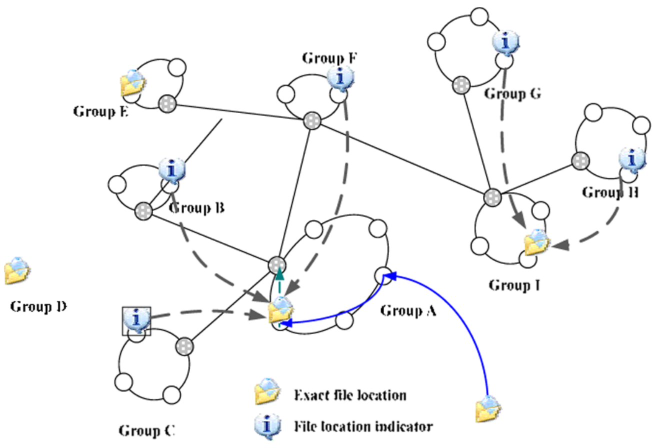
B3 Peer n_r then broadcasts the NEW_FILE notification information and information of the newly arrived file to its neighbouring root peers

B4 When other root peers receive the notification information sequentially, these root peers will notify their responsible sub-peers to get file content and continue notifying other neighbouring root peers.

File insertion operations of our system will replicate files to all the groups of the system. To provide different numbers of file replicas, various replication mechanisms were proposed. In the traditional replication mechanisms proposed by previous works such as PAST and OceanStore systems, all rely on the global information of the system. For obtaining a number of replicas of each file, the replication mechanism must collect the global information of the system by travelling all peers of the system. However, in a fully decentralised peer-to-peer environment, it is difficult to maintain the complete global information owing to the highly dynamic change environment. Therefore, we propose a simple replication mechanism, which is introduced by an example.

A file *f* will be given a replication level *L,* which means the file will be replicated every *L* travelled groups. Peer that should hold the file originally but is skipped by replication level will hold a simple indicator to the peer that really holds the file. Figure 4 is a simple example for $L = 2$.

Figure 4 Examples of replication level $L = 2$ (see online version for colours)



From Figure 4, we can see that initially a file has been uploaded into the group *A*. Then, the file will be replicated every other 2 groups away. Therefore, this inserted file will be replicated in group *D*, *E* and *I*. By using this replication mechanism, we not only increase the file availability but also reduce the storage space requirements.

The algorithm of the file retrieve process is described as the following example.

C1 First, if a client peer *c* wants to retrieve a file f_j, it first contacts with one peer within the selected group and uses the Chord framework to connect with the peer n_i which is responsible for storing the file f_j.

C2 Second, the peer n_i may only contain the file indicator owing to the replication mechanism.

C3 Third, the client peer *c* will follow the indicator to connect with the storage peer that maintains the exact file objects.

C4 Finally, if n_i is found being busy, it will send a notification information to redirect the client peer *c* to other peers that also cache the file f_j.

The storage server peers are usually placed in the same spot permanently and the client peers are usually in the neighbourhood. If a client peer can have a cache, which records the connection route between it and the remote storage peers, it will help to decrease the number of lookup hops. Therefore, in our system, we design the file cache mechanism to achieve load balance among peers and avoid the hotspot problem. The steps of the file cache mechanism are listed as follows.

D1 We will set the threshold of the cache-creation and cache-remove of each peer.

D2 We will record the download frequency of each file of each peer.

D3 When we find a peer is busy, its file download rate will exceed the upper bound of this peer. In such a situation, the transient file will be replicated to its predecessor peer.

3.3 *The process of peers' churn*

Any peer in the system may join or depart freely at any time by itself. Therefore, peers need to check whether the replication system is still maintained well. For the files that do not exist on the peer (it means this peer only has indicator), peer will check to see if the target peer is alive and maintains the pointed file. If not, two situations need to be considered. First, the remote target peer is still available, but the files do not exist. In such a case, we ask the following peers of the original target peer to find another new remote peer. Second, the remote peer is not available. This may be occurred since peers depart or fail. In this situation, the peer that holds the failed indicator will broadcast the new point peer to the peers that hold the files exactly. For files that exist on the peers, peers will check the groups in range $L/2$ (*L* stands for replication level) and remove any redundant files in this range. This process can save the storage space of the system.

Any storage peer that joins or leaves will affect the data location that will lead to the situation of a large amount of files migration. The situation may also cause the failure of lookup. Therefore, the affected peers will first create the *reference* to avoid the missing of locating data by dealing with the change of topology.

4 The system analysis

In this section, we will present a simple quantitative analysis for the characteristics and performance of the overlay of our system.

4.1 Theoretical approximation for travelling hops

In the Malugo system, we referred the inequality of distance d_{int} for any two groups, which was proposed by Zhang et al. (2004), as follows.

$$d_{\text{int}} < \log_M N_{\text{grp}} + 3$$

where M is the average number of neighbours of each group, and N_{grp} is the number of groups. In the system, we consider the average distance at the level of $\log_M N_{\text{grp}} + 1$. From our experience, we can simply substitute 2 to M. Therefore, we can obtain a final equation:

$$d_{\text{int}} = (1/2)\log_2 N_{\text{grp}} + 1.$$

On the other hand, the average hop number for looking up the corresponding nodes of the intra-group operation d_{ira} is mentioned in the Chord system and listed here

$$d_{\text{ira}} = (1/2)\log_2 N_{\text{ira}}$$

where N_{ira} is the average number per group. Therefore, the average locating distance from the bootstrap node to the corresponding node can be considered as follows.

$$d = d_{\text{int}} + d_{\text{ira}} = (1/2)(\log_2 N_{\text{grp}} + \log_2 N_{\text{ira}}) + 1.$$

4.2 The splitting-group decision analysis

On the basis of the previous grouping criteria, the size of different groups may be very large or very small. The size of the group will affect our system performance when we want to search for a specified peer in the local group. Therefore, we first define the expected lookup time (t) for searching for a specific peer.

$$t = \sum \frac{M_{\text{ira}}}{R_n}$$

where t is the travelled time in the local group, R_n is the access rate between {the newly coming peer | client c} and peer n is the travelled hops in the group, and M_{ira} is the routing cost of crossing each peer in local group. Peers that clustered into the same group usually have similar access rate to c. Therefore, we can assume that the value of R_n is equal to R. Besides, the average travelling hops with N_{ira} peers are listed.

$$d_{\text{ira}} = (1/2)\log_2 N_{\text{ira}}.$$

Therefore, the expected lookup time (t) for searching for a specific peer can be converted to the equation as follows.

$$t = \sum \frac{M_{\text{ira}}}{R} \times (1/2)\log_2 N_{\text{ira}}.$$

If we divide this large group into two minor groups, the equation of the expected lookup time (t) for searching for a specific peer can be further represented as follows.

$$t = \sum \frac{M_{\text{ira}}}{R} \times (1/2)\log_2 (N_{\text{ira}}/2) + (1/2)\frac{M_{\text{ira}}}{R}.$$

Since the local group has been divided into two groups, the hop number, which compares with the average number, will be less than half when peers travel in the local group. However, when peers travel among different groups, the hop number, which compares with the average number, will be more than half. Therefore, there is no difference of travelling time in case of splitting the large group or not. According to the above-described statement, activation for merging small groups will not benefit our system. This is because it will be less advantageous to our system for splitting and merging groups and will introduce much extra overhead. Therefore, we do not consider the issue of splitting or merging groups in our system.

5 Simulation results

To evaluate our system, we have implemented a simulator and performed several experiments.

In the following simulation, all peers are distributed in an imaginary map. The speed between any two peers is an exponential distribution from 8 kbps to 8000 kbps. The response time between any two peers is from 3 ms to 400 ms. For measuring the extra maintenance or query cost, we assume that the length of an inter-group message is 512 bytes whereas the length of an intra-group message is 1024 bytes. In addition, the length of a keep-alive message is 128 bytes. In each simulation, we repeat to generate overlay 100 times and get the average results. One of the various goals of our system is to reduce the traffic cost such as travelled hops or travelled time. We will compare these metrics with the CFR system and specify these simulation results.

5.1 System performance with different group bound

This experiment shows the performance of our system under different grouping bound. We vary the grouping bound value from 0 kbps to 4000 kbps in the environment where 2^{12} peers connect with each other. We randomly generated 100 client peers and each client peer randomly performs 10,000 queries simultaneously. From Figures 5–7, we can conclude that with larger grouping bound, client peers not only spend fewer hop numbers but also take less time to reach the proper storage server and achieve higher access rate.

The access rate between storage peers can be increased to improve the maintenance overhead. In addition, we can observe that the results of the travelled time and the access rate, which trend to more smooth, when the grouping bound goes larger than 1300–1700 kbps. This is because that larger grouping bound will create more groups, new joining peers cannot locate to the most proper group.

In Figure 6, the speed line represents the average access rate of client peers. The *spd2NxPr* line is the average access rate to the next peer in local group by using our system topology. The *spd2Neis* line is the average access rate to a peer in its neighbouring group. In addition, the baseline is the average access rate by using the Chord framework. From the simulation, we can obtain a proper value of grouping bound of about 1300 kbps. This value will help us to easily set up our experiment in real world.

Figure 5 Performance evaluation results with different grouping bound (see online version for colours)

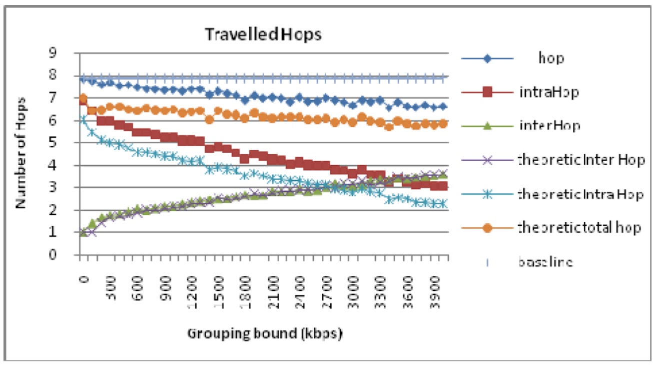


Figure 6 Performance evaluations of the travelled time of different grouping bound

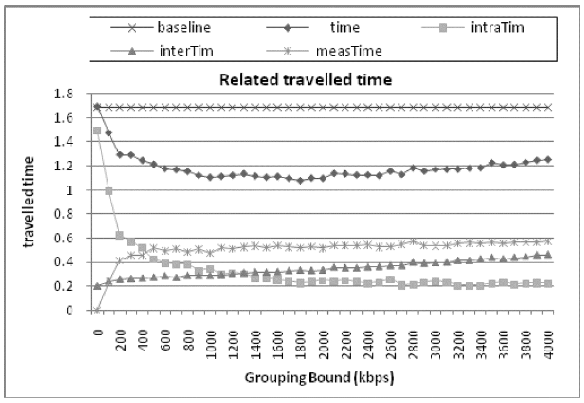
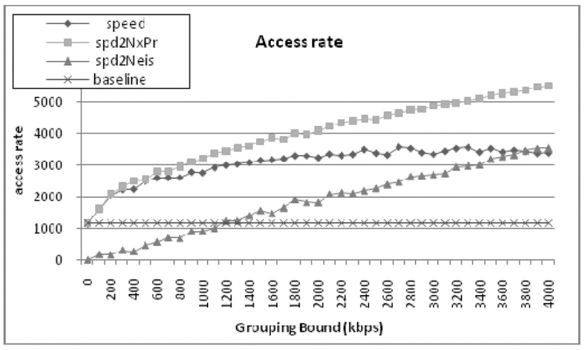


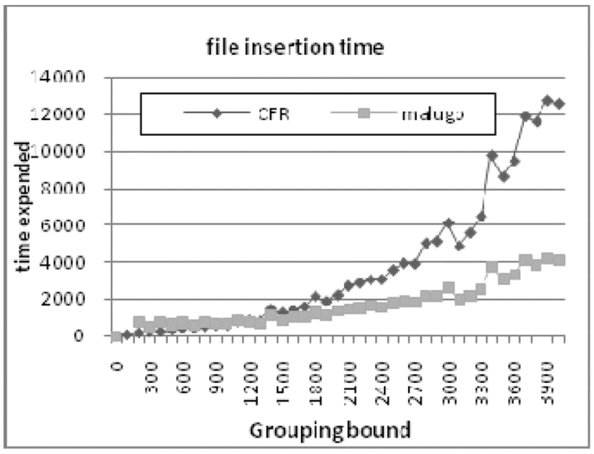
Figure 7 Performance evaluations of the access rate of different grouping bound



We also compare our system with the CFR system in terms of the file insertion time. In our system, we divide the file insertion operations into the inter-group and intra-group routing and obtain the responsible peer for inserting the exact file. A comparison result of the file insertion time is shown in Figure 8.

In this simulation, we insert a file with the 10 Mbytes size to both these systems. The CFR system will create the same number of regions with the number of groups that Malugo has created. From the simulation results, we can realise that when the grouping bound is larger than 1200 kbps, the cost of file insertion of the Malugo system is slighter than the CFR system. However, when the grouping bound is smaller than 1200 kbps, our system introduces more overhead than the CFR system.

Figure 8 Comparisons of file insertion time between our system and the CFR system



5.2 *System performance under different numbers of nodes*

This simulation shows the number of travelled hops under different numbers of storage peers in our system. We vary the number of storage peers from 128 to 65,536 with group bound, which sets to {100, 500, 1000, 2000, 3000, 4000} kbps. In this experiment, 100 client peers will be randomly generated and given a random object to lookup 10,000 times in 100 different network environments.

Figure 9 shows the average number of groups, which system will generate when the number of peers increases. More groups will be generated to handle the join operation of more new peers to achieve higher performance and balance the load of the root peers. Figure 10 shows the relative travelled time. From Figure 10, we can see that when the number of peers increases exponentially, the time increases linearly.

Besides, if we need higher grouping bound, we must increase the number of peers. We first formulate our performance improvement model by evaluating the travelled time here.

$$R_t = \frac{\frac{T - T_b}{T_b}}{\frac{G - G_b}{G_b}}$$

where R_t is the performance improvement under grouping bound B. T is the travelled time under grouping bound B. T_b is the travelled time under the baseline group bound $b = 100$ kbps. Also, G is the number of groups under grouping bound B and G_b is the number of groups under the baseline group bound b. If the performance improvement from b to B in the environment of large number of peers is higher than that of small number of peers, we need a dynamic mechanism to create extra groups to handle the increasing number of peers.

Figure 9 The average number of groups with different numbers of storage peers (see online version for colours)

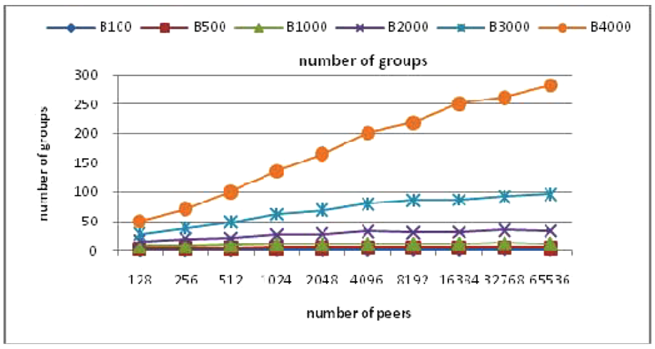
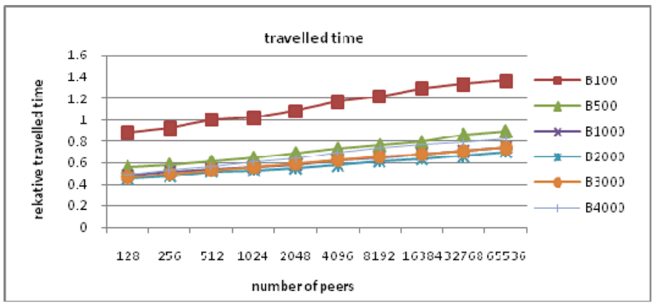


Figure 10 Performance evaluation results of travelled time with different numbers of peers (see online version for colours)



5.3 Evaluation with peer dynamics

In this simulation, we evaluate the dynamic property of the Malugo system. We first set up the environment with peer number $N = 1000$ and set the grouping bound B to 1000 kbps and we also divide peers into about 9.5 groups. The number of successor list used by the local group r was set to 20 and the stabilisation period was set to be 30 s. For comparison, the join and voluntary leave are modelled by a Poisson process with a mean arrival rate R. From Stoica et al. (2003), the expected number of timeouts experienced by a lookup operation in local group can be formulated as follows

$$\frac{R'}{\frac{R'}{l_{\text{ira}}}+\frac{N_{\text{ira}}}{90\log N_{\text{ira}}}+\frac{1}{\log N_{\text{ira}}}}$$

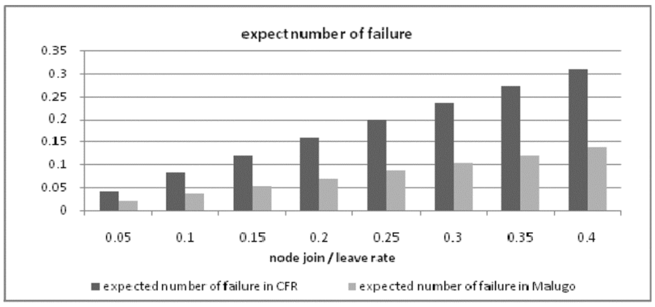
where l_{ira} is the average travelled hops and N_{ira} is the average number of peers per group in local group. Also, $R = R/N_g$ is the arrival rate of join and leave. N_g is the number of groups. A failure occurred during the inter-group locate process can be presented as $l_{\text{int}} \times R \times N_{\text{grp}}/N$. We can obtain our expected number when a lookup operation reach the lookup timeout in local group or failed to locate the proper group as follows

$$\frac{R/N_{\text{grp}}}{\frac{R/N_{\text{grp}}}{l_{\text{ira}}}+\frac{N_{\text{ira}}}{90\log N_{\text{ira}}}+\frac{1}{\log N_{\text{ira}}}}+\frac{l_{\text{int}}\times R\times N_{\text{grp}}}{N}.$$

From Figure 11, we can see that under peer dynamics situation, our system performs better than CFR system. We have fewer expected failure number than CFR system.

Through the above-presented simulation results, we have shown that our system not only reduced the hop numbers but also took less time to search the desired files.

Figure 11 The overlay maintenance overhead under different node join/leave rate



5.4 Experimental results

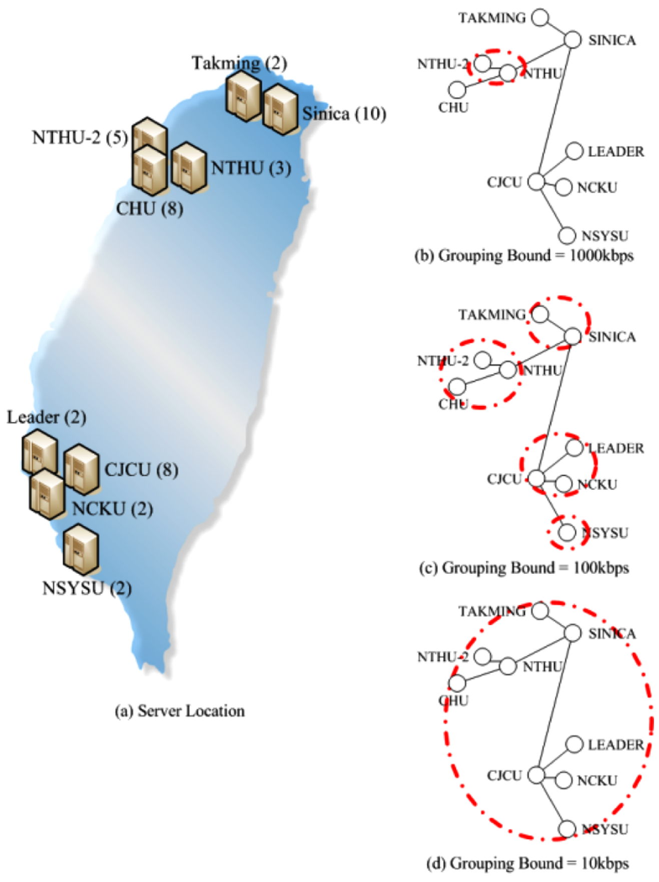
To evaluate the real performance of our system, we have deployed the implemented system on Taiwan UniGrid (2006). The Taiwan UniGrid is a grid platform for researchers in Taiwan to perform distributed researches. We execute the Malugo storage system on 9 sites with 42 servers in 4 cities as shown in Figure 12(a).

We first select the top 10 download files from SourceForge.net as our test data. Then, we use three different grouping bound {1000 kbps, 100 kbps, 10 kbps} to cluster these storage nodes. We set up the storage nodes by setting different grouping bound. Consequently, we run the client peers, which are randomly distributed around these storage nodes to retrieve these test data. Then, we measure the access rate while downloading randomly selected files.

Figure 12(b)–(d) shows that we have successfully clustered physically closer peers together under specific group bound. In the experiment, we found that the average time for measuring the bandwidth between the newly joining peer and the target peer could be less than 10 ms.

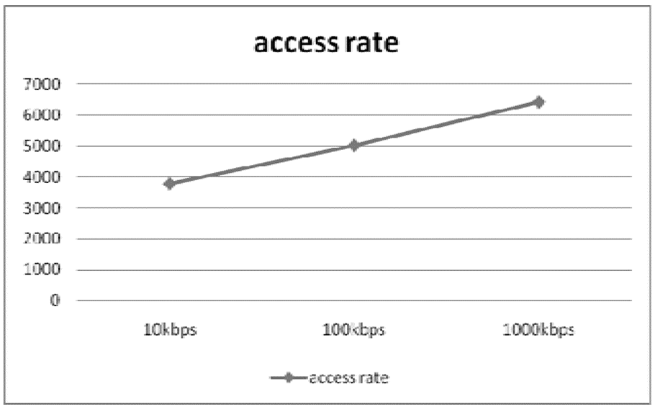
In addition, the average time of locating operation can be less than one second. As a result, we can say that the group operation could be ignored.

Figure 12 (a) Server location of test-bed in Taiwan UniGrid; (b) server group result with group bound being 1000 kbps; (c) server grouping result with group bound being 100 kbps and (d) server group result with group bound being 10 kbps (see online version for colours)



From Figure 13, we can observe that the access rate of download increases while we increase the group bound. Until now, we have shown that our system has been implemented successfully in real world and the performance was better than some previous systems.

Figure 13 Performance evaluation under different grouping bound



6 Conclusions

We have presented the Malugo system, a peer-to-peer storage system that was designed for large-scale collaborative projects. The Malugo system can cluster peers by routing locality automatically, partitioning files to different peers to achieve load balancing and replicating files to different groups to achieve geographical properties without the global information.

In a real world, the Malugo system has been implemented, and we have showed that our system is not only workable but also has excellent performance. Although the Malugo system is an efficient storage platform now, we still go on developing new several features, such as user authorisation and user/file permissions. As a further target, we will develop the Malugo system into a storage platform with file system properties.

References

Clarke, I., Sandberg, O., Wiley, B. and Hong, T.W. (2000) 'Freenet: a distributed anonymous information storage and retrieval system', *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, CA, USA, July, pp.25, 26.

Dabek, F., Kaashoek, M.F., Karger, D., Morris, R. and Stoica, I. (2001) 'Wide-area cooperative storage with CFS', *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, December, Banff, Alberta, Canada, Vol. 35, pp.202–215.

Druschel, P. and Rowstron, A. (2001) 'PAST: a large-scale, persistent peer-to-peer storage utility', *Proceedings of the 8th Workshop on Hot Topics in Operating Systems*, 20–22 May, Germany, pp.75–80.

Gupta, I., Birman, K., Linga, P., Demers, A. and Renesse, R.v. (2003) 'Kelips: building an efficient and stable P2P DHT through increased memory and background overhead', *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS 2003)*, 20–21 February, CA, USA, pp.160–169.

Jin, H., Wang, C. and Chen, H. (2005) 'Boundary Chord: a novel peer-to-peer algorithm for replica location mechanism in grid environment', *Proceedings of the 8th International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN 2005)*, 7–9 December, Las Vegas, Nevada, USA, pp.262–267.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C. and Zhao, B. (2000) 'OceanStore: an architecture for global-scale persistent storage', *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, Cambridge, Massachusetts, USA, Vol. 35, pp.190–201.

Lin, M.R., Lu, S.H., Ho, T.H., Lin, P. and Chung, Y.C. (2007) 'CFR: a peer-to-peer collaborative file repository system', *Proceedings of the 2nd International Conference on Grid and Pervasive Computing*, 2–4 May, Paris, France, pp.100–111.

Liu, Y., Xiao, L., Liu, X., Ni, L.M. and Zhang, X. (2005) 'Location awareness in unstructured peer-to-peer systems', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 16, No. 2, pp.163–174.

Muthitacharoen, A., Morris, R., Gil, T.M. and Chen, B. (2002) ‘Ivy: a read/write peer-to-peer file system’, *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, Massachusetts, USA, Vol. 36, pp.31–44.

Rowstron, A. and Druschel, P. (2001) ‘Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems’, *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, November, Heidelberg, Germany, pp.329–350.

Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F. and Balakrishnan, H. (2003) ‘Chord: a scalable peer-to-peer lookup protocol for internet applications’, *IEEE/ACM Transactions on Networking*, Vol. 11, No. 1, pp.17–32.

Taiwan UniGrid (2006) Obtained through the internet: http://www.unigrid.org.tw/

Xu, Z., Min, R. and Hu, Y. (2003) ‘HIERAS: a DHT based hierarchical P2P routing algorithm’, *Proceedings of the International Conference on Parallel Processing (ICPP)*, October, Kaohsiung, Taiwan, pp.187–194.

Zhang, X.Y., Zhang, Q., Zhang, Z., Song, G. and Zhu, W. (2004) ‘A construction of locality-aware overlay network: mOverlay and its performance’, *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 1, January, pp.18–28.