Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

PipeCF: a DHT-based Collaborative Filtering recommendation system*

SHEN Rui-min (申瑞民), YANG Fan (杨 帆), HAN Peng (韩 鹏), XIE Bo (谢 波)

(*Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200030, China*)

E-mail: {rmshen, fyang, phan, bxie}@sohu.com

Received Dec. 30, 2003; revision accepted Apr. 11, 2004

Abstract: Collaborative Filtering (CF) technique has proved to be one of the most successful techniques in recommendation systems in recent years. However, traditional centralized CF system has suffered from its limited scalability as calculation complexity increases rapidly both in time and space when the record in the user database increases. Peer-to-peer (P2P) network has attracted much attention because of its advantage of scalability as an alternative architecture for CF systems. In this paper, authors propose a decentralized CF algorithm, called PipeCF, based on distributed hash table (DHT) method which is the most popular P2P routing algorithm because of its efficiency, scalability, and robustness. Authors also propose two novel approaches: significance refinement (SR) and unanimous amplification (UA), to improve the scalability and prediction accuracy of DHT-based CF algorithm. The experimental data show that our DHT-based CF system has better prediction accuracy, efficiency and scalability than traditional CF systems.

Key words: Collaborative Filtering, Distributed hash table, Significance refinement, Unanimous amplification
doi:10.1631/jzus.2005.A0118 **Document code:** A **CLC number:** TP391.7

INTRODUCTION

Since David *et al*.(1992) published the first account of using Collaborative Filtering (CF) for information filtering, CF has proved to be one of the most successful techniques in recommendation systems. Its key idea is that users will prefer those items that people with similar interests prefer, or even that dissimilar people do not prefer. According to different techniques used, CF algorithms can be divided into memory-based algorithms and model-based algorithms. Breese *et al*.(1998) empirically analyzed the above two kinds of CF algorithms. Our work is based on memory-based CF algorithms which are the most popular CF algorithms up to now. The main process can be separated into three steps as addressed by Herlocker *et al*.(1999): (1) Similarity weight: Weigh all users with respect to similarity with the active user whose preferences are to be predicted; (2) Selecting neighborhoods: Select those users used to make prediction; (3) Rating normalization and prediction making: Normalize and calculate the weighted sum of selected users' ratings, then make prediction based on that. Herlocker *et al*.(1999) presented an algorithmic framework for performing CF.

Resinck *et al*.(1994)'s GroupLens was the first CF algorithm to automate prediction and used a memory-based algorithm. Like most memory-based algorithms, GroupLens need to compute across the whole user database to calculate the similarities between active user and other users to make prediction. Upendra and Pattie (1995) only used those neighbors whose correlations were greater than a given threshold to make prediction. This approach not only reduced the calculation complexity but also improved the performance. By choosing top-*N* users with the highest correlations, the same improvement can also

* Project (No. 60372078) supported by the National Natural Science Foundation of China

be obtained. However, all the other users' similarities still have to be calculated and its complexity increased quickly both in time and space as the record in the database increases.

Basically, there are two ways to reduce this calculation complexity. The first one is using a model-based algorithm which first constructs certain mathematical models, such as Bayesian Network, Bayesian Classifiers etc. to describe the users and/or their ratings, then learns these models from the database and uses them to make prediction. However, these approaches also needed complex calculation when compiling models and also require a central database to keep all the user data which is not easy to achieve sometimes because of technical reasons and privacy reasons.

The second way is to implement CF in a decentralized way. In fact, as Peer-to-peer (P2P) gains more and more popularity, some researchers began to consider it as an alternative architecture for reducing the calculation complexity (Amund, 2001; Olesson, 2003; Canny, 2002). When implementing CF in a distributed way, the originally centralized user database should be maintained in a distributed way which means that each peer will only keep a fraction of the user database; and that the prediction should be made locally. When making prediction for a particular user, the needed record should first be retrieved from the user's own database and calculated locally. In order to do this, the following two problems have to be addressed: (1) How to store the user database distributed efficiently so that the needed information can be found efficiently; (2) How to identify those records needed to make prediction for a particular user and fetch them efficiently, as retrieving all other users' votes back is not only unreasonable but also unnecessary.

Our main contributions are:

(1) We propose a novel distributed hash table (DHT) based technique for implementing efficient user database management and retrieval in decentralized CF system;

(2) We propose a heuristic algorithm for fetching similar users from DHT overlay network and do recommendation locally;

(3) We propose two novel approaches: significance refinement (SR) and unanimous amplification (UA), to improve the performance of our DHT-based CF algorithm.

The rest of this paper is organized as follows. In Section 2, several related works are presented and discussed. In Section 3, we introduce the architecture and key features of our DHT-based CF system. Two techniques: SR and UA are proposed to improve the scalability and prediction accuracy of DHT-based CF algorithm. In Section 4, the experimental results of our system are presented and analyzed. Finally we make a brief concluding remark and touch on future work in Section 5.

RELATED WORK

Memory-based CF algorithm

Generally, the task of CF is to predict the votes of active users from the user database which consists of a set of votes v_{ij} corresponding to the vote of user i on item j. Memory-based CF algorithm calculates this prediction as a weighted average of other users votes on that item by using the formula:

$$P_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} \varpi(a,j)(v_{i,j} - \overline{v}_i) \tag{1}$$

where $P_{a,j}$ denotes the prediction of the vote for active user a on item j, and n is the number of users in user database. $\overline{v}_i$ is the mean vote for user i as expressed by:

$$\overline{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j} \tag{2}$$

where I_i is the set of items on which user i has voted. The weights $\varpi(a, j)$ reflect the similarity between active user and users in the user database. κ is a normalizing factor to make the absolute values of the weights sum up to unity.

Most memory-based algorithms use Eq.(1) to make prediction and differ only on the ways they calculate the weights. Two most used metrics to calculate the similarities between users are:

1. Pearson correlation coefficient

Pearson correlation coefficient was first introduced into Collaborative Filtering as a weighting method in the GroupLens project. The correlation

between user a and i is:

$$\varpi(a,i)=\frac{\sum_j (v_{a,j}-\overline{v}_a)(v_{i,j}-\overline{v}_i)}{\sqrt{\sum_j (v_{a,j}-\overline{v}_a)^2 \sum_j (v_{i,j}-\overline{v}_i)^2}} \quad (3)$$

where the summation is calculated over those items for which both users a and i have voted.

2. Vector similarity

Vector similarity was first used to measure the similarity between two documents, with each viewed as a vector of word frequency; and their similarity was computed as the cosine of the angle between these two vectors. In Collaborative Filtering, we treat each user record as a document and their votes as frequency of items. So the weights can now be calculated as:

$$\varpi(a,i)=\sum_j \frac{v_{a,j}}{\sqrt{\sum_{k\in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k\in I_i} v_{i,k}^2}} \quad (4)$$

Existing peer-to-peer system

In recent years, more and more P2P systems have been deployed on the Internet. Among all these applications, three main classes of P2P applications have emerged: parallelizable, content and file management, and collaborative.

1. Parallelizable

Parallelizable P2P applications split a large task into smaller sub-pieces that can be executed in parallel over a number of independent peer nodes. Most implementations of this model have focused on computation-intensive applications. The general idea behind these applications is that idle cycles from any computer connected to the Internet can be leveraged to solve difficult problems that require extreme amounts of computation. Most often, the same task is performed on each peer using different sets of parameters. Examples of implementations include code breaking, portfolio pricing, risk hedge calculation, market and credit evaluation, and demographic analysis.

2. Content and file management

Content and file management P2P applications mainly focus on how to store information on and retrieve information from various peers in the network. The model that popularizes this class of application is the content exchange model. Applications like Napster and Gnutella allow peers to search for and download files, which are primarily music files. In recent years, the foundations of P2P file systems have been explored by a number of research projects. In addition to file sharing, these applications focus more on Collaborative Filtering techniques that build searchable indices over a P2P network. Technologies like JXTA Search can be used in conjunction with P2P file sharing application like Gnutella to enable more up-to-date searches over a large, distributed body of information.

3. Collaborative

Collaborative P2P applications allow users to collaborate, in real time, without relying on a central server to collect and relay information. Instant messaging is one subclass of this class of application. Services, such as Yahoo! Messenger, AOL, and Jabber instant messaging, have become more and more popular among Internet users. Similarly, shared applications, which allow people (e.g., business colleagues), who are possibly thousands of miles apart, to interact while viewing and editing the same information simultaneously, are also emerging. Examples include Buzzpad and distributed PowerPoint. Games are a final type of collaborative P2P application. P2P games are hosted on all peer computers and updates are distributed to all peers without requiring a central server. Example games include NetZ 1.0 by Quazal, Scour Exchange by Centers pan, Descent, and Cybiko.

DHT routing algorithms

Unfortunately, the initial designs for P2P systems have significant scaling problems; for example, Napster has a centralized directory service, and Gnutella employs a flooding-based search mechanism that is not suitable for large systems.

In response to these scaling problems, several research groups proposed a new generation of scalable P2P systems that support distributed hash table (DHT) functionality, such as CAN (Ratnasamy *et al*., 2001), Chord (Stocal *et al*., 2001), Pastry (Rowstron and Druschel, 2001), and Tapestry (Zhao *et al*., 2001). In these DHT systems, each file is associated with a key (produced, for instance, by hashing the file name) and each node in the system is responsible for storing a certain range of keys. There is one basic operation in

these DHT systems, *lookup*(*key*), which returns the identity (e.g., the IP address) of the node that stores the object with the certain key. This operation allows peers to store and retrieve files based on their keys, thereby supporting the hash-table-like interface.

The core of these DHT systems is the routing algorithm. A DHT overlay network is composed of several DHT nodes, with each node having some other nodes as its neighbors. When a *lookup*(*key*) is issued, the lookup request will be routed through the overlay network to the node, which stores the file with that key.

Each of the proposed DHT systems listed above –Tapestry, Pastry, Chord, and CAN–employ a different routing algorithm. All of them take, as input, a key and, in response, route a message to the node responsible for that key. The keys are strings of digits of some length. Nodes have identifiers, taken from the same space as the keys (i.e., same number of digits). Each node maintains a routing table consisting of a small subset of nodes in the system. When a node receives a query for a key for which it is not responsible, the node routes the query to the neighbor node that makes the most "progress" towards resolving the query. The notion of progress differs from algorithm to algorithm, but in general is defined in terms of some distance between the identifier of the current node and the identifier of the queried key.

The primary goals of DHT are to be an efficient, scalable, and robust P2P routing algorithm aimed to reduce the number of P2P hops which are involved when we locate a certain file, and to reduce the amount of routing state that should be preserved at each peer. In Chord (Stocal *et al*., 2001), each peer keeps track of information on log$N$ other peers (N is the total number of peers in the community). When a peer joins and leaves the overlay network, this highly optimized version of DHT algorithm will only require notifying log$N$ peers about that change.

OUR DHT-BASED CF APPROACH

Architecture of DHT-based CF system

The main difference between our DHT-based CF system and traditional centralized CF system is that both the maintenance of user database and the complex computation task of making prediction are done in a decentralized way. Each user keeps his votes locally. The system generates a unique key for each particular <*ITEM_ID*, *VOTE*> tuple of each user by hashing it. So each user will have M keys, with M being the number of items he has rated. These keys are then used to construct a DHT overlay network as described in Section 2.3. When a user wants to look up other similar users which have the same particular <*ITEM_ID*, *VOTE*> tuple, it can fetch them from the DHT overlay network efficiently. So with the DHT overlay network, all the users in the CF system are connected together and can find their wanted similar neighbors efficiently through a DHT routing algorithm. Fig.1 gives the architecture of our DHT-based CF system.

Basic DHT-based CF algorithm

On the basis of the decentralized storage of user votes, we introduce our decentralized CF algorithm, called DHT-based CF algorithm, shown in Fig.2.

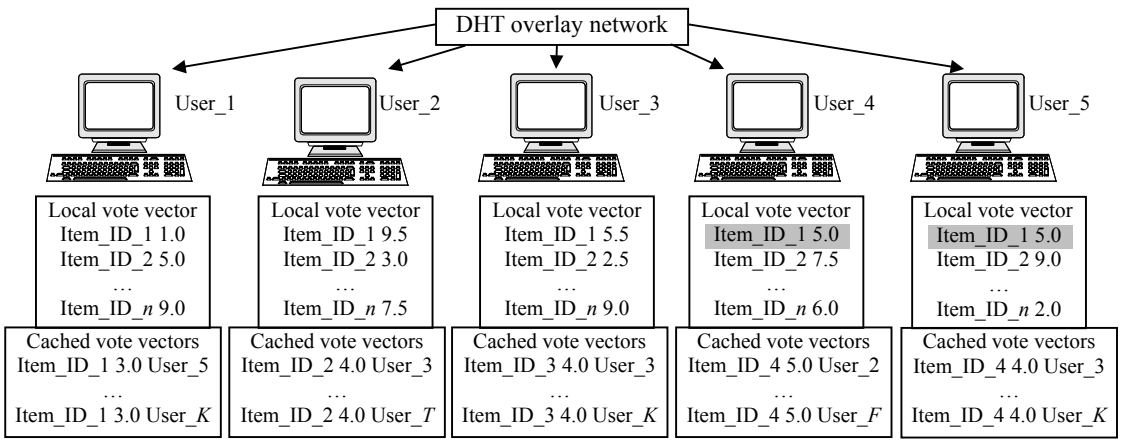

Fig.1 Architecture of DHT-based CF system

Algorithm: DHT-based Collaborative Filtering
Input: training set, test set, a target item and a given top *K* selection
Output: mean absolute error of prediction
Method:

(1) Construct a DHT overlay network (we simulate its two main functions: *put*(*key*) and *lookup*(*key*)).

(2) Every user of training set hashes <*ITEM_ID*, *VOTE*> as a key and put all the keys to the DHT overlay network by calling *put*(*key*) function.

(3) Each instance of the test set fetches similar neighbors from the DHT overlay network by calling *lookup*(*key*) function. All fetched users' votes construct a local training set LOCAL_TRAINING_SET.

(4) Each instance of the test set searches the top *K* neighbors from LOCAL_TRAINING_SET and computes the corresponding prediction for the target item.

(5) Compute the mean absolute error of prediction and output.

Fig.2 DHT-based Collaborative Filtering

There are two key pieces to our decentralized CF system: the lookup mechanism used to locate similar users and fetch their actual rating. The decentralized storage (and hence decentralized retrieval) in decentralized CF system makes the CF calculation inherently scalable (every user do recommendation locally instead of depending on a centralized server); the hard part is finding the similar peers from which to retrieve the actual rating. We devise a scalable solution to the problem of locating similar users in decentralized CF system, i.e., give a user vote vector; we can find the IP address of the node(s) which is similar to the user. By using a DHT-based routing algorithm, our solution can achieve:

(1) Scalability: it must be designed to scale to several million nodes.

(2) Efficiency: similar users should be located with reasonable speed and low overhead in terms of the message traffic generated.

(3) Balanced load: in keeping with the decentralized nature, the total resource load (traffic, storage, etc.) should be roughly balanced across all the nodes in the system.

The neighbor choosing strategy of DHT-based CF algorithm is based on the heuristic principle that people with similar interests rate at least one item with similar votes as we can see in Fig.5. So we only select similar users in the subset in which users have the same <*ITEM_ID*, *VOTE*> tuple. The key idea of our algorithm is hashing every user for every rated item. Our DHT-based CF algorithm includes two main DHT functions: *put*(*key*) and *lookup*(*key*), and Figs.3 and 4 illustrate them respectively.

Algorithm: DHT-based CF puts a peer *P*'*s* vote vector to DHT overlay network
Input: test set (*P*'s vote vector)
Output: NULL
Method:

(1) *P* generates a unique 128-bit DHT key K_{local} (i.e. hash the system unique username).

(2) *P* hashes one <*ITEM_ID*, *VOTE*> tuple to key *K*, and routes it with test set (*P*'s vote vector) to the neighbor P_i whose local key K_{i_local} is the most similar to *K*.

(3) When P_i receives the PUT message with *K*, it caches it. And if the most similar neighbor is not itself, it just routes the message to its neighbor whose local key is most similar to *K*.

(4) For each rated item, *P* repeats Steps 2 and 3.

Fig.3 DHT-based Collaborative Filtering Put

Algorithm: DHT-based CF looks up similar users for a peer *P*
Input: test set (*P*'s vote vector)
Output: training set (retrieved similar users vote vectors)
Method:

(1) *P* generates a unique 128-bit DHT key K_{local} (i.e. hash the system unique username).

(2) *P* hashes one <*ITEM_ID*, *VOTE*> tuple to key *K*, and routes it with test set (*P*'s vote vector) to the neighbor P_i whose local key K_{i_local} is most similar to *K*.

(3) When P_i receives the LOOKUP message with *K*, if P_i has enough cached vote vectors with the same key *K*, it returns the vectors back to *P*, otherwise it just routes the message to its neighbor whose local key is most similar to *K*. Anyway, *P* will finally get similar users and the corresponding vote vectors for key *K*.

(4) For each rated item, *P* repeats Steps 2 and 3. Then *P* outputs all the received similar users' vote vectors.

Fig.4 DHT-based Collaborative Filtering Lookup

DHT-based CF Put algorithm is used to construct a DHT overlay network and fill data in it. DHT-based CF Lookup algorithm is used to lookup and fetch similar users with the same <*ITEM_ID*, *VOTE*> tuple in order to construct a local training set to make recommendation. The main purpose of Steps 2 and 3 in Fig.3 is to make every peer in the DHT overlay network keep several buckets which contain a group of users with the same <*ITEM_ID*, *VOTE*> tuple, from which the Lookup algorithm can fetch similar users later in its Steps 2 and 3.

Extensions to memory-based algorithm

1. Significance refinement (SR)

In the basic DHT-based CF algorithm, we return all users which have the same <*ITEM_ID*, *VOTE*> tuple as that of the active user and find that the algorithm has an $O(N)$ fetched user number (N is the total user number) as Fig.7 shows. In fact, as Breese presented in (Breese *et al*., 1998) by the term inverse user frequency, universally liked items are not as useful as less common items in capturing similarity. So we introduce a new concept significance refinement (SR) which reduces the returned user number of the basic DHT-based CF algorithm by limiting the number of returned users for each <*ITEM_ID*, *VOTE*> tuple. We term the algorithm improved by SR as Return *K* which means "for every item, the DHT-based CF algorithm returns no more than *K* users with the same <*ITEM_ID*, *VOTE*> tuple". The experimental result showed that this method reduced the returned user number dramatically and improved the prediction accuracy.

2. Unanimous amplification (UA)

Enlightened by the method of case amplification (Breese *et al*., 1998) which emphasizes the contribution of the most similar users to the prediction by amplifying the weights close to 1, we argue that we should give special award to the users who rated some items with the same vote by amplifying their weights, which we term unanimous amplification. We transform the estimated weights as follows:

$$w'_{a,i}=\begin{cases} w_{a,i}, & N_{a,i}=0 \\ w_{a,i}\alpha, & 0<N_{a,i}\le\gamma \\ w_{a,i}\beta, & N_{a,i}>\gamma \end{cases} \tag{5}$$

Where $N_{a,i}$ denotes the number of items for which user a and user i have the same votes. In our experiment, typical value for α was 2.0, for β was 4.0, and for γ was 4. Experimental result showed that the UA approach improved the prediction accuracy of both the traditional and DHT-based CF algorithms.

EXPERIMENTAL EVALUATION

In this section, we describe the dataset, metrics and methodology for the comparison between traditional and DHT-based CF algorithm, and present the results of our experiments.

Dataset

We used EachMovie dataset (EachMovie, 1997) to evaluate the performance of the improved algorithm. The EachMovie dataset was provided by the Compaq System Research Center, which ran the EachMovie recommendation service for 18 months to experiment with a Collaborative Filtering algorithm. The information they gathered during that period consisted of 72 916 users, 1 628 movies, and 2 811 983 numeric ratings ranging from 0 to 5. To speed up our experiments, we only used a subset of the EachMovie dataset which contains 500–5000 users.

Metrics and methodology

The metrics for evaluating the accuracy of a prediction algorithm can be divided into two main categories: statistical accuracy metrics and decision-support metrics. Statistical accuracy metrics evaluate the accuracy of a predictor by comparing predicted values with user-provided values. Decision-support accuracy measures how well predictions help user select high-quality items. We use Mean Absolute Error (MAE), a statistical accuracy metrics, to report prediction experiments as it is most commonly used and easy to understand:

$$MAE=\frac{\sum_{a\in T}|v_{a,j}-p_{a,j}|}{|T|} \tag{6}$$

where $v_{a,j}$ is the rating given to item j by user a, is the predicted value of user a on item j, T is the test set, $|T|$ is the size of the test set.

We selected 2000 users and chose one user as active user per time and the remaining users as his candidate neighbors, because every user only recommended himself locally. We used the mean prediction accuracy of all the 2000 users as the system's prediction accuracy. For every user's recommendation calculation, our tests were performed using 80% of the user's ratings for training, with the remainder being used for testing.

Experimental result

We designed several experiments for evaluating our algorithm and analyzing the effect of various

factors (e.g., SR and UA, etc.) by comparison. All our experiments were run on a Windows 2000 based PC with Intel Pentium 4 processor having a speed of 1.8 GHz and 512 MB of RAM.

1. The efficiency of neighbor choosing

We used a dataset of 5000 users and show among the users chosen by DHT-based algorithm, how many are in the top-100 users most similar to the active users which will be chosen by traditional CF algorithms in Fig.5. We can see from the data that when the total user number rises above 1000, more than 80 users who are most similar to the active users are chosen by our DHT-based algorithm.

2. Performance comparison

We compared the prediction accuracy of traditional CF algorithm and DHT-based CF algorithm while we apply both top-all and top-100 user selection on them. The results are shown as Fig.6 showing that the DHT-based algorithm has better prediction accuracy than the traditional CF algorithm. From the result, we find that by eliminate those users who have a high co-relation between the active users but no same ratings, the prediction accuracy can be increased.

3. The effect of significance refinement

We limited the number of returned user for each <*ITEM_ID*, *VOTE*> tuple to 2 and 5 and do the experiment. By which we mean that for each <*ITEM_ID*, *VOTE*> tuple, our algorithm will return at most 2 or 5 users. The user for each <*ITEM_ID*, *VOTE*> tuple was chosen randomly. The result of the number of user chosen and the prediction accuracy are shown in Fig.7 and Fig.8 respectively. The result shows that:

(1) "Return all" had an $O(N)$ returned user number and its prediction accuracy was also not satisfying;

(2) "Return 2" had the least returned user number but the worst prediction accuracy;

(3) "Return 5" had the best prediction accuracy and the scalability was still reasonably good (the returned user number was still limited to a constant as

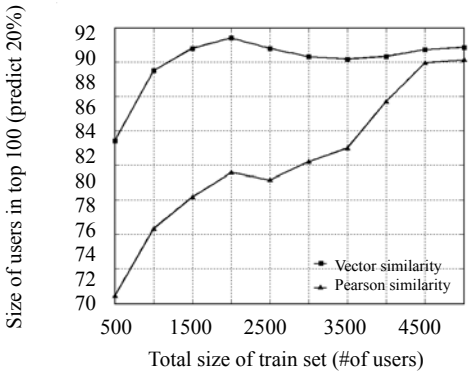

Fig.5 How many users of DHT-based CF which have same <*ITEM_ID*, *VOTE*> tuple are in the traditional CF's top 100

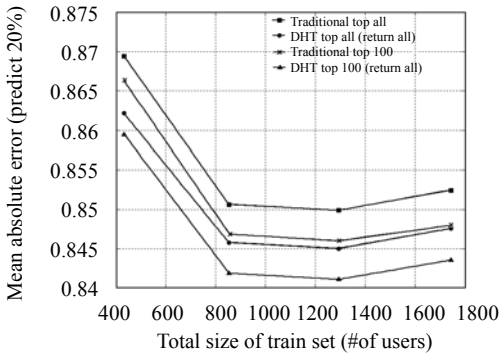

Fig.6 DHT-based CF vs traditional CF

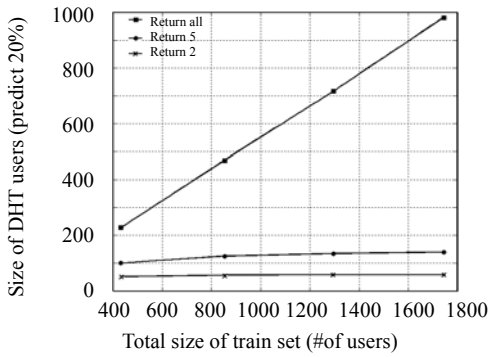

Fig.7 The Effect on scalability of SR on DHT-based CF algorithm

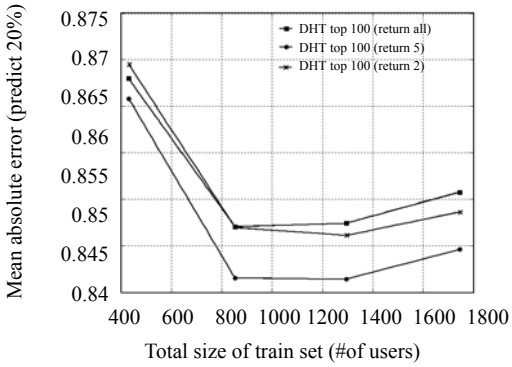

Fig.8 The Effect on prediction accuracy of SR on DHT-based CF algorithm

the total user number increased).

4. The effect of unanimous amplification

We adjusted the weights for each user by using Eq.(5) while setting the value for α as 2.0, β as 4.0, γ as 4 by our experience and did the experiment again. We used the top-100 and "Return all" selection method. The result showed that the UA approach improved the prediction accuracy of both the traditional and the DHT-based CF algorithm. From Fig.9 we can see that when the UA approach is applied, the two kinds of algorithms have almost the same performance.

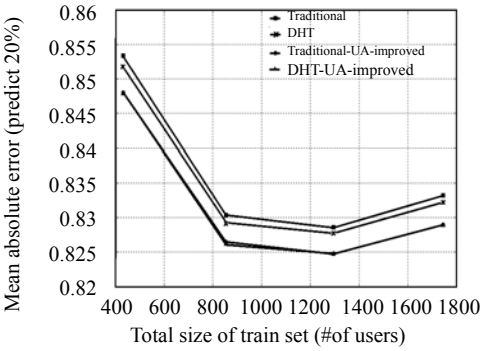

Fig.9 The effect on prediction accuracy of unanimous amplification

CONCLUSION

In this paper, we propose a novel decentralized CF system by using distributed hash table (DHT) technique to implement efficient user database management and retrieval. Then we propose a heuristic algorithm to fetch similar users from the DHT overlay network and do recommendation locally. Finally, we propose two novel approaches: significance refinement (SR) and unanimous amplification (UA) to improve the performance of our DHT-based CF algorithm. The experimental data showed that our DHT-based CF system had better prediction accuracy, efficiency and scalability than traditional CF systems.

Our future work includes investigation on a more efficient decentralized user database management and *K*-Nearest Neighbor (KNN) methods which can dynamically self-organize users (Wang, 2002; Krämer and Schmidt, 2001) with similar interests in combining content-based filtering techniques. We would also like to investigate the influence of parameters choosing in UA.

References

Amund, T., 2001. Peer-to-peer Based Recommendations for Mobile Commerce. Proceedings of the First International Mobile Commerce Workshop, ACM Press, Rome, Italy, p.26-29.

Breese, J., Heckerman, D., Kadie, C., 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, p.43-52.

Canny, J., 2002. Collaborative Filtering with Privacy. Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, p.45-57.

David, G., David, N., Brian, O.M., Douglas, T., 1992. Using Collaborative Filtering to weave an information tapestry. *Communications of the ACM*, **35**(12):61-70.

EachMovie, 1997. EachMovie Collaborative Filtering Data Set. http://research.compaq.com/SRC/eachmovie.

Herlocker, L.J., Konstan, A.J., Borchers, A., Riedl, J., 1999. An Algorithmic Framework for Performing Collaborative Filtering. Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, p.230-237.

Krämer, B.J., Schmidt, H.W., 2001. Component and tools for on-line education. *European Journal of Education*, **36**(2):14-41.

Olesson, T., 2003. Bootstrapping and Decentralizing Recommender Systems, Licentiate Thesis. Department of Information Technology, Uppsala University and SICS, Uppsala, Sweden.

Resinck., P., Neophytos, I., Mitesh, S., Peter, B., John, R., 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. Proceedings of the 1994 ACM conference on Computer Supported Cooperative Work, Chapel Hill, North Carolina, United States, p.175-186.

Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S., 2001. A Scalable Content-addressable Network. ACM SIGCOMM, San Diego, CA, USA, 2001.

Rowstron, A., Druschel, P., 2001. Pastry: Scalable, Distributed Object Location and Routing for Large Scale Peer-to-peer Systems. IFIP/ACM Middleware, Hedelberg, Germany.

Stocal, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H., 2001. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. ACM SIGCOMM, San Diego, CA, USA, p.149-160.

Upendra, S., Pattie, M., 1995. Social Information Filtering: Algorithms for Automating "Word of Mouth". Proceedings of the SIGCHI Conference on Human factors in Computing Systems, Denver, Colorado, United States, p.210-217.

Wang, F., 2002. Self-organizing Communities Formed by Middle Agents. Proceedings of the First International Conference on Autonomous Agents and Multi-agent Systems, Bologna, Italy, p.1333-1339.

Zhao, B.Y., Kubiatowicz, J.D., Joseph, A.D., 2001. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. Tech. Rep. UCB/CSB-0-114, UC Berkeley, EECS.