A Distributed Collaborative Filtering Recommendation Model for P2P Networks

Jun Wang[1], Jian Peng[1,2], and Xiaoyang Cao[3]

[1] School of Computer Science, Sichuan University, Chengdu 610065, China
[2] Department of Computer Science, University of Maryland, MD 20742, USA
[3] Chengdu University of Information Technology, 610225, China

Abstract. Conventional collaborative filtering(CF) recommendation applies the user-based centralized architecture. This architecture has some problems of sparsity and scalability, in addition to not fit the current popular P2P architecture. Therefore, this paper proposes a distributed model to implement the CF algorithm by maintaining the user's record information distributedly in each nodes throughout the network, constructing a DHT, applying the Chord algorithm to realize locating of the record and designing the corresponding communication policy to obtain data needed.

Keywords: collaborative filtering, probabilistic relevance model, distributed, Chord.

1 Introduction

With the rapid development of network technology and the continuous increase of resources, users need a more personalized service to help users to find the information they need from enormous resources. The so-called personalized service considers the need and choice of individual users, applies different strategies for different users and meets the users' personalized needs by providing pertinent content. The most prevailing and effective way is to use collaborative filtering(CF). CF is based on the fact that, if different users' rating of the information which has already been judged is similar, then their rating of other information which has not been judged should be similar. Thus, basing on other users' evaluation and combining target user's personal history record, the collaborative filtering recommendation system provides the user with the information he may interest most.

In the past, most of the collaborative filtering systems search and maintain relevant information in a central server. This centralized architecture has two major problems. One is data sparsity. In e-commerce or other applications, the number of items and users is enormous, therefore, even the active users only rate a very small part of the items and even the hot items are only rated by a very small part of the total users. In such a sparse circumstance with large amount of data, it's difficult to judge the similarity of different users. Another problem is the system's scalability. The time complexity of user-based Collaborative filtering algorithm, in the worst-case scenario, is the O(MN),where M denotes the

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 1–10, 2009.

number of users and N denotes the number of items. Thus, with the increase of users and items, the amount of computation will dramatically increase. Besides that, P2P networks have become a popular way of sharing resources and, taking into account of system robustness and issue about intellectual property rights, pure P2P architect does not include a central server or central router to control the entire network, Such as Gnutella, Freenet file system. Therefore, this paper proposes a distributed collaborative filtering recommendation model applicable to the P2P architecture.

2 A Distributed Collaborative Filtering Recommendation Architecture

Conventional collaborative filtering recommendation system is as shown in Figure 1. The central server runs an application to response the request from the clients. When receiving a client's request, the server searches the data needed from the user data set and the item data set, sends the data to CF module, gets the recommendation set through the recommendation algorithm set by the system and finally sends the result back to the client. This centralized approach makes the process simple and has a high efficiency when the number of users and items is not very great. However, such a centralized structure is not applicable to P2P network, besides the problems of the data sparsity and poor scalability. Some studies propose the cluster model. By measuring the similarity of users, the cluster model divides the user base into different clusters and each cluster is a collection of the most similar users. Its purpose is to assign the user to the cluster containing users most similar to itself. By calculating the similarity only with the users in the same cluster, thus it greatly reduces the dimensions of the user-item matrix and greatly improves the scalability and the performance of collaborative filtering algorithm.[7] However, the grouping for the user base is not very precise, largely affecting the recommendation's accuracy. If applying more precise cluster in order to improve the accuracy, expenses produced will largely offset the improvement brought by the decrease in dimensions of user-based matrix[4]. Therefore, by storing the data

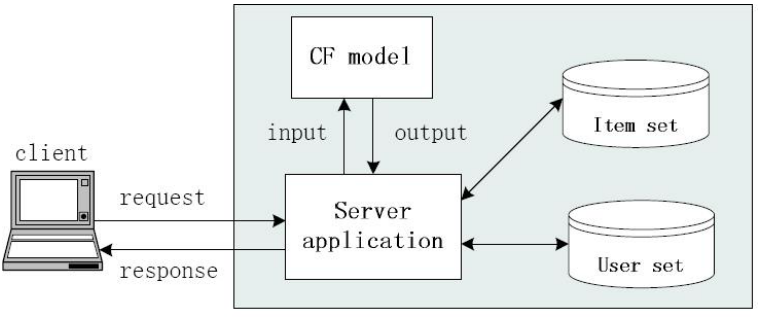

Fig. 1. A centralized CF architecture

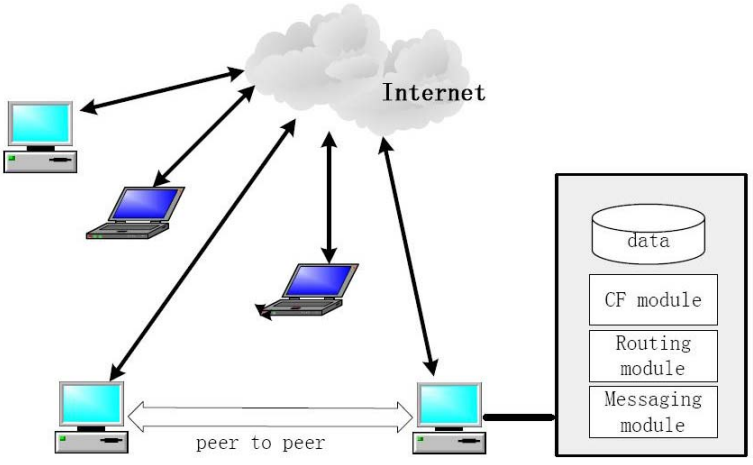

Fig. 2. A distributed CF architecture

and performing computing in each node, this paper proposes a distributed CF recommendation system architecture.

Unlike the centralized architecture, in the distributed architecture as shown in Figure 2, data is scattered throughout the network and the corresponding computing is also performed in each node. All the nodes in the system are linked together orderly in accordance with a certain rule, and each node shares records it maintains. The target node in the system needs to collect the necessary data to calculate the corresponding recommendation set, so every node needs to achieve the following four modules.

User data module: user data module keeps user's own downloading record, which is a collection of movie names the user has downloaded. This record not only reflects the tendency of the user's preference, but is also part of data source which other users use to calculate its own user-item relevance.

CF module: When the target collects all the necessary data, as the input source, the probabilistic relevance model is used to calculate the relevance between the user and the item. And according to the relevance, return 15 items with the highest relevance as the recommendation set for the target user.

Routing module: Compared to a centralized server, which all the data is stored in a central database server, in the distributed architecture, user's data is maintained by each node alone. Since it needs to search and locate other users' record when calculating the recommendation set for target user, routing module uses the chord algorithm to build a distributed hash table(DHT) in the whole system with each node responsible for maintaining one small part, and to provides a highly effective way to achieve the locating for the necessary records.

Message processing module: message processing module is mainly to handle the inter-node message, responsible for the registration and deregistration of nodes' records, the corresponding work for requesting data from other nodes or sending data to other nodes, thus achieve the objective that the data can be shared in all nodes.

2.1 CF Module

CF module is the core of the whole system. By inputting relevant data to calculate the user-item relevance, obtain the recommendation aiming at the user's preference. In this paper, we use probabilistic relevance model proposed in [9] to build the user-item relevance model . Originally, the probabilistic relevance model is used for information retrieval, judging the relevance between the query and the document. In recommendation system, the user's history record represents the user's personal preference, so we can regard the user's history record as the model's input, and introduce two random variables r (relevant) and $\overline{r}$ (irrelevant). Assume there are N items, denoted as I_a(a=1,2,,N),and M users, denoted as U_b(b=1,2,,M). the relevance of item for user is denoted as follow

$$R_{I_a,U_b} = log\frac{P(r|I_a,U_b)}{P(\overline{r}|I_a,U_b)} \tag{1}$$

$P(r|I_a,U_b)$ denotes the relevance probability between I_a and U_b, $P(\overline{r}|I_a,U_b)$ denotes irrelevance probability between I_a and U_b. Factorize $P(r|I_a,U_b)$ with $P(U_b|I_a,r)P(r|I_a)/P(U_b|I_a)$

$$R_{I_a,U_b} = log\frac{P(r|I_a,U_b)}{P(\overline{r}|I_a,U_b)} = log\frac{P(U_b|I_a,r)}{P(U_b|I_a,\overline{r})} + log\frac{P(r|I_a)}{P(\overline{r}|I_a)} \tag{2}$$

To simplify the model without affecting its generality, we only care about the positive relevance and remove the irrelevance terms. Then the equation will become as follow:

$$R_{I_a,U_b} \propto logP(U_b|I_a,r) + logP(r|I_a) \tag{3}$$

The items that the user have interacted with(downloading, watching or browsing) before reflect the positive interest of user for those items. Let L_b denote the downloading list of U_b. If I_a is a member of the downloading list of U_b, then $L_b(I_a)$=1, otherwise $L_b(I_a)$=0. Therefore, the item set the downloading list keeps can represent the user's preference and can be used to calculate the relevance as the input source. Assuming each item in downloading list is irrelevant, the equation(3) becomes:

$$R_{I_a,U_b} \propto \sum_{\forall I_i:I_i\in L_b} logP(I_i|I_a,r) + logP(r|I_a) \tag{4}$$

Apply the Bayers in equation(4):

$$R_{I_a,U_b} \propto \sum_{\forall I_i:I_i\in L_b} log\frac{P(r|I_a,I_i)}{P(r|I_a)} + logP(r|I_a) \tag{5}$$

As the equation(5) shows, calculation of the item-user relevance is converted into calculation of the probability of an item and the relevance probability between items. If a user likes item A and item B at the same time, a certain correlation is established between them. Therefore we can calculate the probability

of the relevance between two items, namely $P(r|I_a, I_i)$, through the frequency of those two items appearing in the same downloading list and calculate the probability of an item, namely $P(r|I_a)$, through the frequency of this item in the downloading list as follow.

$$P(r|I_a, I_i) = (\sum_{n=i}^{M} L_n(I_a) \wedge L_n(I_i))/M \tag{6}$$

$$P(r|I_a) = (\sum_{n=i}^{M} L_n(I_a))/M \tag{7}$$

Different from user-based collaborative filtering algorithm, this is a item-based collaborative filtering algorithm. The item-based recommendation model uses users' downloading records to reflect relationship between the items, reducing the impact caused by data sparsity by avoiding calculating the adjacent user set.

2.2 Routing Module

In the distributed environment, user data is all scattered throughout the network. So, before providing users recommendation values, the system must obtain other users' downloading lists from other nodes. Therefore we use the Chord algorithm as system's routing and search algorithm, building a DHT to quickly search and locate nodes. The first P2P routing algorithm, which each node maintains a routing table, randomly stores some other nodes as the neighbor nodes. When needing to search resources in the network, the node sends the query message to every neighbor node stored in the routing table, and then according to the routing information of the neighbor nodes, forward the query message to other nodes in the network alike, which is a broadcasting method to search the nodes needed. This manner is suitable for small and medium-sized networks, but, as the network expands, the searching time will increase dramatically. In addition, the broadcasting message will put burden on the network and, due to TTL, message's searching will be limited to a certain extent, which can not guarantee that the records needed will be found. Therefore, the system uses DHT to locate the nodes the system needs. DHT provides a distributed search service. All the index entries constitute a big hash table, in which every index entry is denoted as a key-value pair, namely, pair(K, V). K denotes keywords, which is the hash of file name or other descriptive information of the file, and V denotes the node's IP address or other descriptive information of the node which actually stores the file. Following certain rules, each participating node maintains part of the whole hash table. Through the query message, Nodes can route to the corresponding node according to the keyword, obtaining all the addresses of the nodes which stores this file. In this way, the changes of the nodes only have a small impact on the network, and the system has good scalability.

In this collaborative filtering model, through building a distributed hash table, we link the item name with the address which have downloaded that item already and achieve the fast locating. In the system, each keyword and node has a m-bit

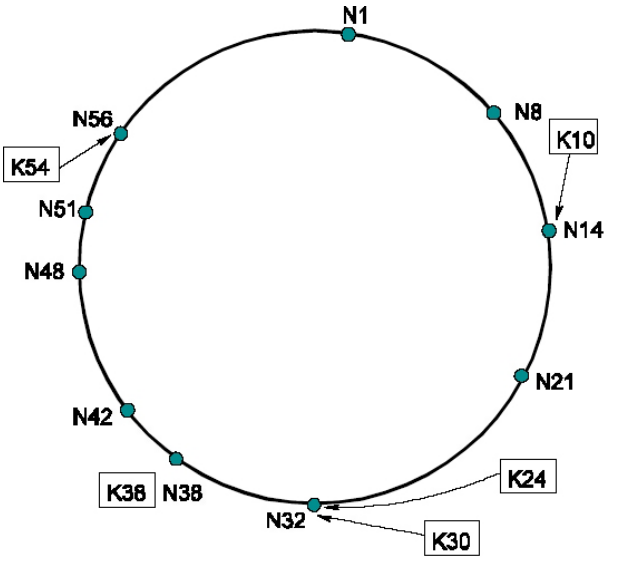

Fig. 3. Chord ring

identifier respectively. Using a hash function SHA-1, hash the keyword (Here is the file name) as the keyword identifiers, and hash node's IP address as the node identifier. All the nodes are arranged along a logical ring clockwise in ascending order according to its node identifier. The item whose keyword identifier is equal to K, is indexed in the node whose identifier is equal to K or this node's successor. The successor node in the Chord ring is the node nearest to and bigger than node K clockwise, denoted as successor (K)[3]. As shown in Figure 3, the item with identifier 10 stores at the node with identifier 14, the items with identifier 24 and 30 stores at the node with identifier 32. Each node contains a reference to its successor node, when search the file name, the query message is transferred through the successor node along the ring until arriving at the corresponding nodes indexing the record.

In order to speed up searching speed, routing algorithm uses the extended algorithm to achieve O(logN) search performance. Each node needs to be responsible for the maintenance of a routing table known as the finger table. For The node n, the ith reference in its finger table points to the node m which is at lest 2^{i-1} greater than node n in the Chord ring, m = successor (n $+2^{i-1}$).The fist entry in the finger table points to the direct successor of node n in the ring, that is, successor (n +1). If node n's finger table does not contain successor(k), node n finds the node f which is closest to successor(k)in the finger table and sends a query request to node f. Node f proceeds the search in the same way and this process will continue, recursively, until arriving the node in accordance with keyword k.

Since node joining/exiting have an impact on the system, routing algorithm also establishes a policy to overcome the impact of system changes. When a new node joins the network, firstly the new node n calculates the node identifier

according to IP, through the node identifier finds its successor node and inserts it between its successor nodes and successor's previous precursor nodes. At the same time, due to system change, part of the index entries originally maintained by its successor will be transferred to the hash table that the new node maintains, and then will be removed in the successor. Similarly, when node leaves, in addition to need to inform the successor node and precursor node re-points its precursor and successor respectively, it needs to transfer the index entries it maintains to its successor.

2.3 Messaging Module

The routing algorithm just achieves distributed data search, in order to achieve distributed collaborative filtering algorithm, and ensure the node can access the data needed from other nodes, every node also needs to achieve the corresponding inter-node message processing function for different events.

Event I. registration of downloading records: For collaborative filtering recommendation system, each item in downloading list can be regarded as a shared resource, namely, K in (K, V) pair. Each node maintains a hash table as part of the whole DHT. When new node joins, in addition to transfer corresponding (K,V) entries of the successor node to the new node, the new node needs to share its own downloading records. For each download record, namely file name, it gets the keyword identifier K after being hashed, and locates the corresponding node according to K. The new node traverse each item in the downloading list, sends the request to the corresponding node according to K and creates corresponding index entries in the harsh table on the corresponding node. In this way, the downloading records of the new nodes become a part of DHT, exposed to the entire network. Other nodes can find and access to the downloading records of the new node.

Event II. request for records: In order to calculate preference value between the target user and item, need to obtain item data associated with items in the target user's downloading list, that is, by analyzing other users' downloading lists which also include the item that the target node has downloaded before, calculate the user-item relevance according to the probabilistic relevant model mentioned above. In the entire network, we build a DHT. Due to the property of DHT, the same items in various downloading list are registered and indexed in the same node which is decided by identifier K(the hash of the file name). Therefore, through the node (like node B in Figure 4) in accordance with identifier K, the system can obtain all addresses of nodes (like nodes C, D, E in Figure 4)which have downloaded this item, and through these nodes which have downloaded this item, the system can obtain these nodes' downloading records, which are the basis for calculating the relevance between target item and other items. As shown in Figure 4, a node needs to complete corresponding work in the process of requesting records. Assuming the number of the nodes is n, since Node A can only send message to most n nodes and receive message from most n nodes, the message complexity is O(n)

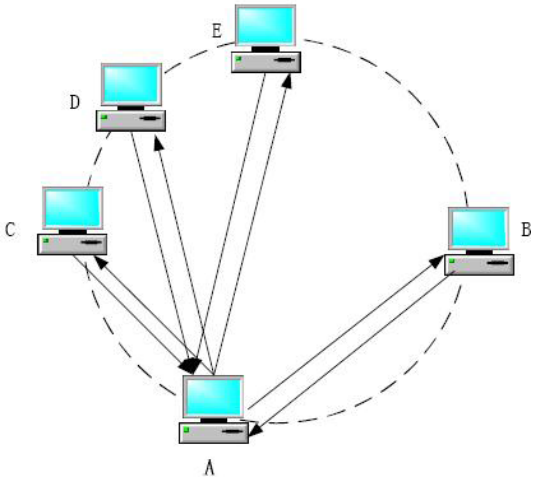

Fig. 4. The process of request record

Algorithm of the protocol is represented as followed:

For each item I_a in node A of user U_1
    For each Node N_1 which has downloaded I_a
        send the data of its downloading list to Node A
    For each item I_b which is in the downloading list of nodes which has downloaded I_a
        compute the relevance between I_a and I_b

For each item I_c which is collected from other nodes mentioned above
    compute the relevance between I_c and U_1
Get the top n relevant items of user U_1

Event III. deregister the downloading records: When the node withdraws, the downloading records maintained by this node in its hash table will not be available. Therefore, when node exits the network, the target node should send deregistration message to all the nodes which index the target node's downloading record. The corresponding nodes will delete the corresponding entry in the hash table it maintains when receiving the message. The algorithm is as follows: when the target node leaves the network, for every entry of the downloading list, hash the item name to obtain the identifier K, send deregistration request to the corresponding node according to K.

3 Experiment

In order to verify the performance of distributed collaborative filtering model, we built a system with 30 nodes, each node maintaining 20 to 30 downloading

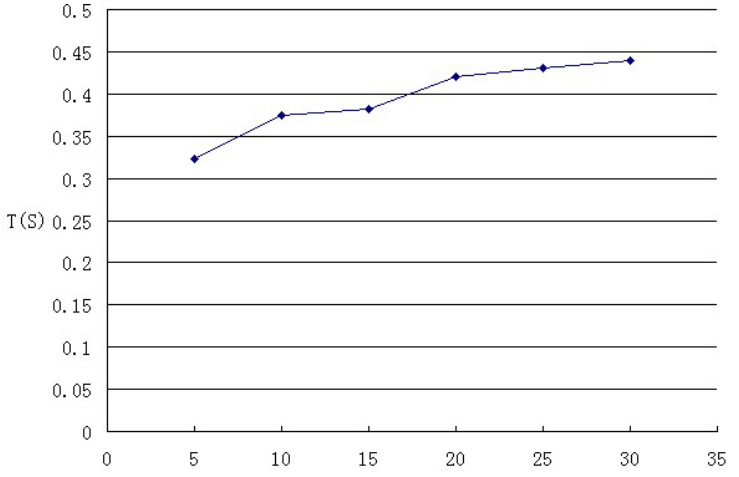

Fig. 5. The experimental result

records. Experiments are divided into six groups. At first, only five nodes participate, and each time increase 5 more nodes to participate in the system. In each experiment, a fixed node is chose to record required time to get the recommendation set. Each group of experiments was conducted five times, calculating the average as experimental result.

As show in figure 5, that node can get the recommendation set in a short period. And with the increase of the number of participating nodes, the time of obtaining recommendation set grows in a low rate, which proves that system has a very good scalability and is applicable to P2P network.

4 Conclusion

Collaborative filtering systems in e-commerce and other applications are more and more widely used. Addressing the sparsity and scalability of the centralized collaborative filtering algorithm, this paper proposes a distributed architecture. By maintaining the user downloading records in various nodes, using Chord as the routing algorithm for data search, designing corresponding inter-node message processing policy to enable the node to access the data needed by searching the DHT, and calculating the user-item relevance according to the equation derived from the probabilistic relevance model, the user can obtains the personalized recommendation set from other users without a central server.

References

1. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Proc. of the tenth international conference on Information and knowledge management, pp. 247–254 (2001)

2. Tveit, A.: Peer-to-peer based recommendation for mobile commerce. In: Proc. of the First International Mobile Commerce Workshop, pp. 26–29 (2001)
3. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proc. ACM SIGCOMM Conf., pp. 149–160 (September 2001)
4. Linden, G., Smith, B., York, J.: Amazon com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing, 76–80 (2003)
5. Balakrishnan, H., FransKaashoek, M., Karger, D., Morris, R., Stoica, I.: Looking up data in P2P systems. Communications of ACM 46(2), 43–48 (2003)
6. Peng, H., Bo, X., Fan, Y., Ruimin, S.: A scalable p2p recommender system based on distributed collaborative filtering. Expert systems with applications, 203–210 (2004)
7. Xue, G., Lin, C., Yang, Q., Xi, W., Zeng, H., Yu, Y., Chen, Z.: Scalable collaborative filtering using cluster-based smoothing. In: Proc. of the 28th Annual Int'l ACM SIGIR Conf., pp. 114–121 (2005)
8. Shakery, A., Zhai, C.: A probabilistic relevance propagation model for hypertext retrieval. In: Proc. of the 15th ACM International Conference on Information and Knowledge Management (CIKM 2006), pp. 550–558 (2006)
9. Wang, J., Pouwelse, J., Lagendijk, R., Reinders, M.R.J.: Distributed Collaborative Filtering for Peer-to-Peer File Sharing Systems. In: Proc. of the 21st Annual ACM Symposium on Applied Computing (2006)