2009 International Conference on Advanced Information Networking and Applications

Introducing Group Participation Support into P2P Web Caching Systems

Akihiro Iwamaru Tsuyoshi Itokawa Teruaki Kitasuka Masayoshi Aritsugi
Computer Science and Electrical Engineering
Graduate School of Science and Technology
Kumamoto University
Kumamoto 860-8555, Japan
{*gangan@dbms., itokawa@, kitasuka@, aritsugi@*}*cs.kumamoto-u.ac.jp*

Abstract

In this paper, we introduce group participation support into decentralized peer-to-peer(P2P) web caching systems. The support allows P2P web caching systems to have more participators, and as a result, realizes highly scalable caching systems. An implementation design of systems is described in this paper. Some experimental results show that every node in our system can get better performance by the introduction of group participation support.

1. Introduction

P2P networks have been applied to such areas as file sharing, instant messaging, audio and video streaming, and web caching. Because P2P networks do not need centralized servers, P2P systems are resilient to failure. A peer can participate in a P2P system autonomously, and is to contribute to as well as to be served from the system.

In particular, web caching systems can benefit from P2P networks. Similar to proxy-based web caching systems, a P2P web caching system can reduce access latency. The storage of a P2P web caching system consists of a set of storages provided by each peer. Thus, the more the number of participating peers grows, the larger the whole capacity of the caching system becomes. In other words, a P2P web caching system can have scalable storage capacity, dissimilar to proxy-based systems.

However, conventional techniques about P2P web caching systems may prevent many computers from participating in the systems, because they did not take account of variations of participation. For example, imagine that there are several idle computers and you would like to participate in a P2P web caching system with the computers. Each computer becomes a peer in a conventional P2P web caching system, and the merit of participating in the system is very limited even though you provide several computers to the system. You may quit participating in the system with the computers, or participate in the system with a single computer. As a result, the number of peers in the system would be made low.

In this paper, we discuss group participation support for developing decentralized P2P web caching systems that allow peers to build a group in the network. Our proposal divides a storage provided by a peer into two fields: one is for the whole P2P network and the other is for the group that the peer belongs to. That is, a peer that belongs to a group contributes to both the whole system and the group. Not to mention, a peer that does not belong to any group can also participate in our system. We discuss how to handle the two kinds of storages in order to realize group participation in a P2P web caching system. Our proposal can keep the hit rate of our system be almost equal to or even outperform that of a conventional system, which cannot support group participation, with the same number of peers. Note that the conventional system may essentially not be able to have the same number of peers because of the lack of group participation support. Note also that in our system the benefit of a node from the P2P caching system would depend on whether the node belongs to a group or not; the introduction of group participation into a P2P caching system results in the fact that the hit rate of nodes in groups outperforms that of nodes that do not belong to any group. We think the fact can meet reasonable situation in the real world.

Also, we report some experimental results. There are generally some skews in web accesses and such skews are often modeled with Zipfian distribution. In the experiments, we modeled that web accesses by all peers in a P2P network were Zipfian distributed and those by peers in a group were also Zipfian distributed, but the Zipfian distribution of web accesses by a group was different from that by another group. This is because we assumed that interests of a peer in a group is somehow similar to that of another peer in the same group but interests of a group are different from those of another group.

The remainder of this paper is organized as follows. Section 2 mentions related work and compares with our proposal. Section 3 proposes group participation support for P2P web caching systems. Section 4 reports some experimental results for evaluating our proposal, and Section 5 concludes this paper.

2. Related work

There have been studies on web caching systems based on P2P technologies. For example, Xiao, Zhang, and Xu [1] proposed a web document sharing technique. Kim and Park [2] studied a P2P proxy cache system. The two systems exploit a P2P network as backup storage for a centralized proxy server. Xu, Hu, and Bhuyan [3] proposed a system consisting of many clients' local caches and the topology of the system is somehow similar to our proposal. However, in the system clients are managed in layers, and some powerful clients manage other clients. On the other hand, our proposal is based on decentralized P2P web caching systems and does not need a centralized server or superclients.

Squirrel [4] is also a decentralized P2P web cache system. The system used Pastry [5] to store and locate web objects and realized a large scalable cache storage consisting of many desktop machines. Sheng and Bastani [6] proposed another decentralized P2P web cache system. However, they did not allow nodes to participate in the system as a group, and we think that they may make some nodes hesitate to participate in the system.

Similar to our proposal, peers with similar interests can be organized into a group in IntraCache [7]. The way of group participation in the system is that a peer in a group participates in the P2P network and the other nodes in the group just dangle to the peer. Similar architecture to IntraCache can be found in some literatures including [8], which supports locality of peers in a wireless mobile P2P network. In their architecture, the representative node in a group is distinguished from the other nodes in the group, and thus we do not think the architecture can realize a really scalable P2P web caching system. On the other hand, any peer in our proposal can contribute to the P2P web caching system and thus our proposal can be really scalable.

3. Proposal

3.1. Topology

To make the discussion in this paper concrete, we use Chord [9] as the base P2P network topology; we think our proposal can be applied to other structured P2P network topologies including Pastry [5], Tapestry [10], CAN [11], and BATON [12]. For simplicity, the number of groups that a node can belong to is limited to one in this paper.

We adapt a multi-ring structure [13] for constructing and managing groups in a P2P network, because it can be managed with ease. Fig. 1 shows an example of our proposal's topology where there are ten nodes in the P2P network, which is drawn by a solid line, and two groups, which are drawn by dotted lines. The finger table of a node that will belong to a group is extended to hold the information of neighbor nodes for managing the ring for

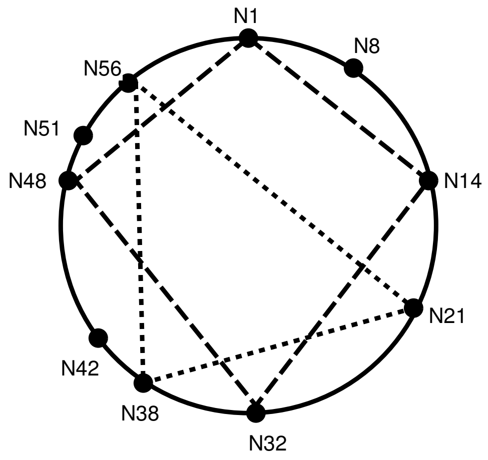

Figure 1. Our proposal's topology.

the group. Needless to say, if another P2P network than Chord is adapted, then the way of holding the information needs to be changed accordingly. When a node joins a group, the node contacts a node in a group and gets the necessary information, and the ring is updated.

We borrow the hash function used in the base P2P network, Chord, for identifying objects and nodes. Xue et al. [14] proposed a new hashing scheme for constructing two layer topology-aware structured overlay network. In our proposal, however, we decided not to adopt the mechanism because we would like to make our system simple and to avoid reducing our proposal's applicability to other P2P networks.

3.2. Storage policy

If a node belongs to a group, the storage that the node provides to the system is divided into two fields: one is for the whole P2P network and the other is for the group that the node belongs to. We call them outer storage and inner storage, respectively, in this paper. Fig. 2 depicts a storage situation of nodes appearing in Fig. 1.

Let o and i be the sizes of the outer storage and inner storage, respectively, we must set $o \gg i$. Note that the storage is used for the system, that is, the storage is used as the outer storage, through the time when the node does not belong to a group. Moreover, if there are empties in one field, then they are used for the other field, if necessary, for space performance.

When a node that does not belong to a group requests a web object and it has not been cached in the system, then the object is retrieved and obtained from the web and is stored in the outer storage of the peer that has the responsibility to manage it in the P2P network. For example, when a peer that does not belong to a group requests objects K38 and

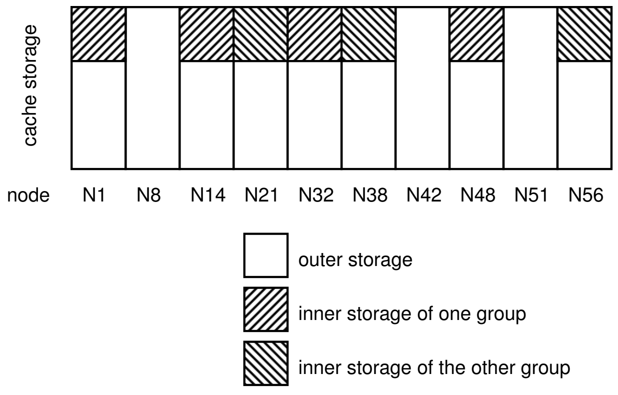

Figure 2. Storage assignment.

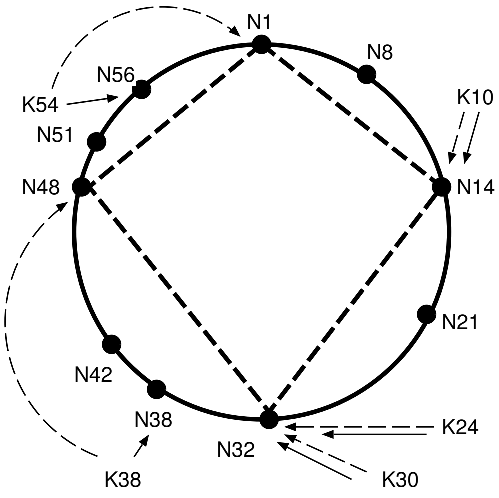

Figure 3. Object locations.

K54 and they are missed and obtained from the web in the case shown in Fig. 3, they are stored in the outer storages of nodes N38 and N56, respectively. On the other hand, when a node that belongs to a group requests a web object and the system misses it, then the object is also retrieved and obtained from the web and is stored in the inner storage of the peer that has the responsibility to manage it as the inner ring were the Chord ring. In Fig. 3, K38 and K54 are stored in N48 and N1, respectively, in this case.

According to this storage policy and lookup policy, which will be described later, a web object can exist twofold in the network if it is cached in inner storage before being cached in outer storage. For example, K38 can be stored in outer storage of N38 and in inner storage of N48 in Fig. 3. However, a web object cannot exist twofold in a node, i.e., in the outer and inner storage of a node in our system. In the case shown in Fig. 3, the nodes that have the responsibility to manage objects K10, K24, and K30 in the P2P network are N14, N32, and N32, respectively, and the nodes also have the responsibility to manage the objects in the inner ring. In this case, if K24, for example, has been cached in inner storage of N32 and a node that does not belong to the group requests the object, then the cached K24 is returned to the node and moves from inner storage to outer storage of N32.

3.3. Replacement policy

Cached objects in the outer and inner storage of a peer are managed in LRU manner independently. If replacement occurs in one field of the storage and some room in the field is used for the other field, then the room is used for the field in the replacement.

3.4. Lookup policy

When a node that does not belong to a group but participates in the network accesses a web object, it checks whether it is stored or not by traversing the outer ring of the system. If the cached object is found incidentally in outer storage of the node during traversing the outer ring, then the object is used and the node updates the LRU information of the outer storage. If the cached object is found in inner storage of the node, then the object is used, the object moves from the inner storage to the outer storage, and the node updates the LRU information of the both storages. If the object is not found, then the node that issued the request retrieves and obtains the object from the web, and also makes the object be cached in the outer storage as mentioned in Section 3.2.

On the other hand, when a node that belongs to a group accesses a web object, the node first checks the existence of the object in inner storages of the peers in the group by traversing its inner ring. If it is found in inner storages, then the cached object is returned and the node updates the LRU information of the inner storage. If not, the node then checks the existence in the outer ring of the system and follows the way described above. Similar to the case described above, if the object is not found, then the object is retrieved and obtained from the web, and the node that issued the request makes the object be cached in the inner storage as mentioned in Section 3.2.

3.5. Discussions

There are several problems to be discussed in our proposal described so far. Here we discuss a couple of them. One is how to implement inner rings. Because we assumed that the number of nodes constructing a group is not so large, a node in an inner ring has just the successor location in the current implementation of our system; if the number grows larger than what we expected, then we should have to implement inner rings more sophisticated for reducing the cost of traversing them and access latency. However, note that let n and g be the number of nodes in the network and a group, respectively, we can naturally assume that $n \gg g$ and thus simple implementation must be enough.

Another is how to implement the policies described in this section. For example, as shown in Fig. 3, there can be a web object cached twofold in our system. If there are a lot of such objects in our system, the performance must be bad since a larger part of spaces are wasted than those without group participation support. We think, however, such situations would not occur so much; in our system, the most popular web objects accessed by nodes in the network and by nodes in a group are cached in outer storages and inner storages, respectively, because of the characteristics of our storage and replacement policies. For example, if there is a web object existing twofold in the network, it has been stored in inner storage and outer storage in this order. Note that the object can be a popular object accessed by nodes in a group and those in the network. Note also that the cached object in inner storage would be naturally replaced in a while because $o \gg i$. As a result, web objects that are very popular among nodes in a group but not among nodes in the network tend to be cached only in inner storage in the group and those that are very popular among nodes in a group and also among nodes in the network tend to be cached only in outer storage in the network.

Another is how to construct groups. In the discussion of this paper, it is supposed that a node of a group knows that it is a member of the group in advance. By introducing group participation into P2P web caching systems, we can take autonomous group creations in the network into account. Anglade, Tiemann and Vignoli [15] presented an approach to automatically create groups with similar preferences of music in P2P networks. We think the approach will become a good tip for considering how to construct groups in our proposal. More details are out of the scope of this paper and will be included in our future work.

4. Simulation

We evaluated our proposal by running simulation developed with Overlay Weaver[16]. In the simulation we compared our proposal with a simple Chord-based P2P web caching technique. The most significant difference between our proposal and the Chord-based P2P web caching technique is whether group participation is supported or not; the Chord-based P2P web caching technique is a decentralized system and the whole storage provided by a peer in the network is dedicated to the caching system.

4.1. Simulation configuration

According to the results shown in [17], we assumed that object accesses were issued by Zipfian method as $f(k, \alpha, N) = \frac{k^{-\alpha}}{\sum_{n=1}^{N} n^{-\alpha}}$ where we set N and α to 2,500,000 and 1.0, respectively. The total number of object accesses was set to 700,000.

According to the results of [18], there should be some skews of object access patterns among groups while assuming the above to the total object access pattern. We set the object access pattern as follows. In the simulation, there were two types of peers: one was to participate in the network and the other was to participate in both the network and a group. Let us assume that there are 100 nodes and two groups, each of which consists of 5 nodes out of them. In order to model the access skews among groups, we numbered each group and set that i-th group accesses j-th popular objects where $i \equiv j (\bmod \lfloor \frac{100}{5} \rfloor)$ in the Zipfian distribution. For example, the two groups access 1st, 21st, 41st, 61st, 81st, 101st, $\cdots$, and 2nd, 22nd, 42nd, 62nd, 82nd, 102nd, $\cdots$ popular objects, respectively. According to the results of [18] again, some popular objects in a group are generally accessed by nodes not belonging to the group. In order to simulate such situation, we set that a popular object in a group was accessed by a node in the group and by a node randomly chosen from the whole nodes in a contiguous two accesses to the object.

For simplicity, we created groups in advance and we assumed that no node left from the network or a group during the simulation, i.e., the numbers of nodes in a group and in the network did not change through the simulation, and the sizes of groups were the same in the simulation; we set the size of a group 5 nodes in the simulation. In addition, each node provided the same size of cache storage which was counted as the number of objects; studies with taking account of object sizes will be included in our future work.

In the following, let `Ca` be the size of cache storage of a node, `Cg` be the size of inner storage out of `Ca` of a node belonging to a group, `Gr` be the ratio of the number of nodes belonging to a group to the numbers of nodes in the network, `Hc` be the hit rate of the conventional environment where the number of nodes is equal to that of our proposal while every node participated in the network as a single peer, `Ha` be the hit rate of our proposal, `Hg` be the hit rate of the nodes belonging to groups in our proposal, and `Hn` be the hit rate of the nodes each of which participated in the network as a single peer in our proposal.

We measured hit rates with varying the numbers of nodes in networks, `Gr`, `Ca`, and `Cg` in the simulation. The first 100,000 out of 700,000 accesses were used for constructing the initial situations where web objects were cached in the system, and the following values were obtained using the rest 600,000 accesses.

4.2. Group participation ratios

For showing the effectiveness of introducing group participation support into P2P caching systems, we measured

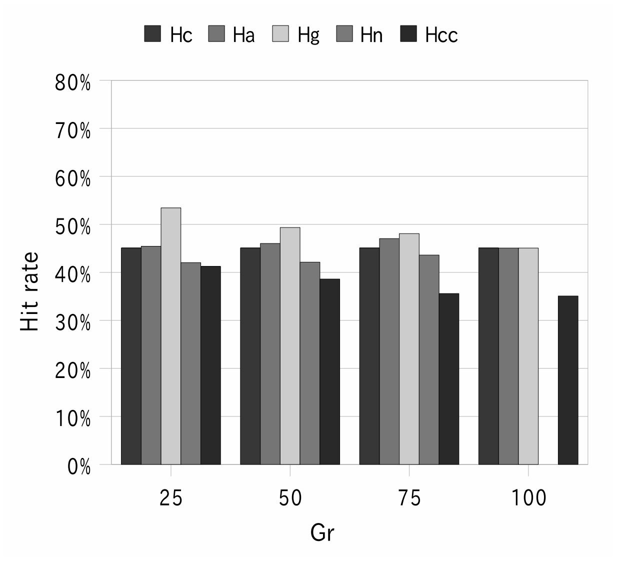

Figure 4. The number of nodes was 100, Ca= 50, and Cg= 5.

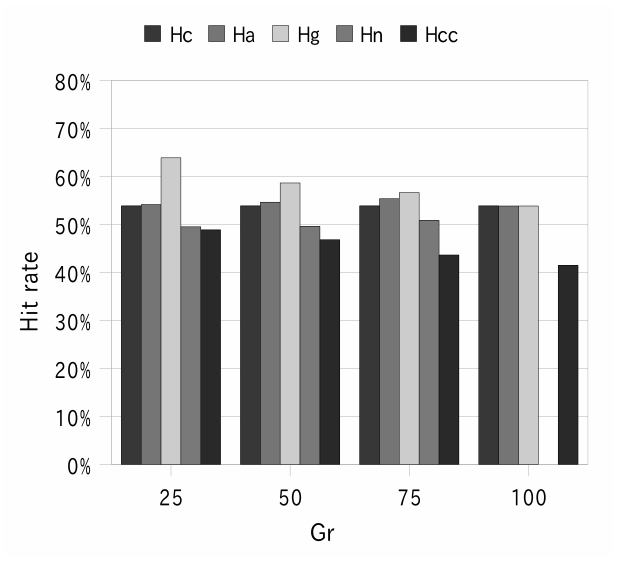

Figure 5. The number of nodes was 300, Ca= 50, and Cg= 5.

hit rates with varying Gr. Figs. 4, 5, and 6 show hit rates measured in cases where we set Ca= 50 and Cg= 5, and varied the number of nodes and Gr. As shown in the figures, Ha is slightly better than Hc in all cases except for Gr= 100%, i.e., group participation can slightly improve the hit rate of the system when the number of groups is moderate in the network. Also, Hg outperforms Hc and Ha, in particular when the number of groups is low. In contrast, Hn is worse than Hc. Note that the number of nodes in the environment where we measured Hc was the same as that in the environment where we measured Ha. As we wrote in Introduction, we may not be able to have the same number of peers in the system if the system could not support group participation.

To compare Hn with the hit rate of the conventional system more fairly, we created networks of nodes, which consisted of nodes that did not belong to a group plus nodes each of which chosen from each group in the networks used for measuring Ha. Thus, the number of nodes in the networks is the number of nodes of single participators plus the number of groups in the corresponding networks used for measuring Ha. We set the web object access patterns of a peer in the two networks identical and measured hit rates, which are shown by Hcc in Figs. 4, 5, and 6. As shown in the figures, Hn outperforms Hcc in all cases, and according to the results we can say that every node in our system can get benefit from supporting group participation.

4.3. Cache capacity

We measured hit rates with varying Ca for examining the influence of the size of cache storage upon the system

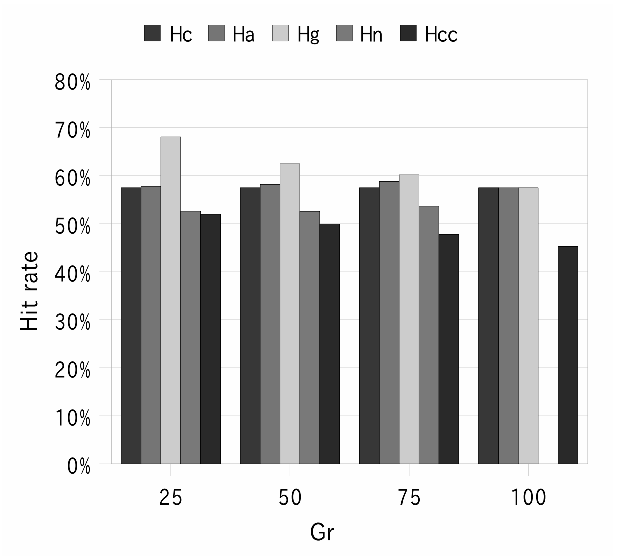

Figure 6. The number of nodes was 500, Ca= 50, and Cg= 5.

performance. Figs. 7, 8, and 9 show hit rates measured in cases where we set Cg= 5 and Gr= 25%, and varied the number of nodes and Ca. As shown in the figures, the more the value of Ca is, the better the hit rates are. On the other hand, the relations among the hit rates shown in the figures look similar in all the three cases. Because the total cache capacities of two cases where Ca= 150 in Fig. 7 and Ca= 50 in Fig. 8 are the same, the Ha's and Hc's are almost the same. However, because the total size of storage used as inner storage are different in the two cases, Hg in

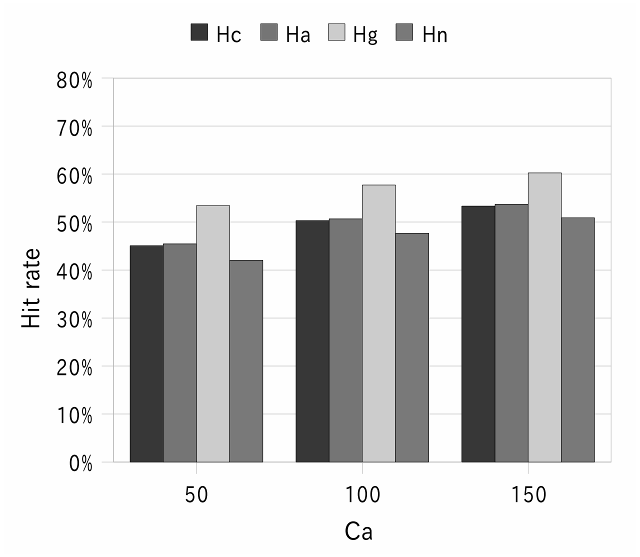

Figure 7. The number of nodes was 100, Cg$= 5$, and Gr$= 25\%$.

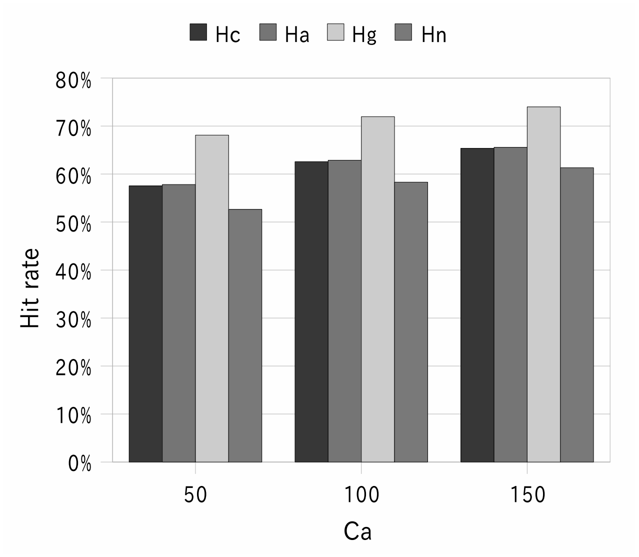

Figure 9. The number of nodes was 500, Cg$= 5$, and tt Gr$= 25\%$.

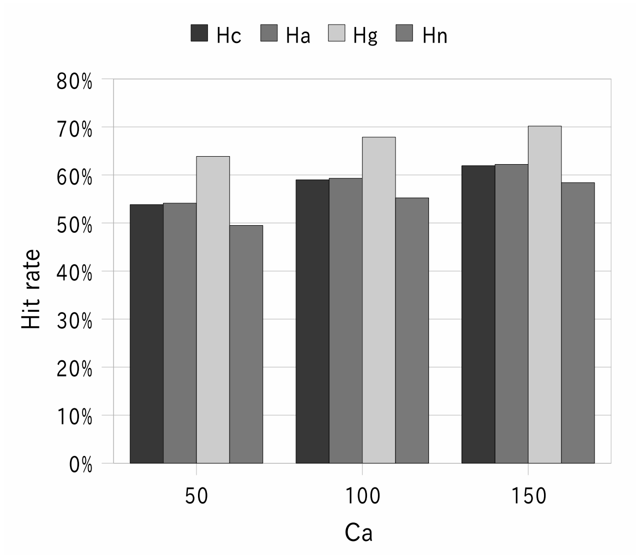

Figure 8. The number of nodes was 300, Cg$= 5$, and Gr$= 25\%$.

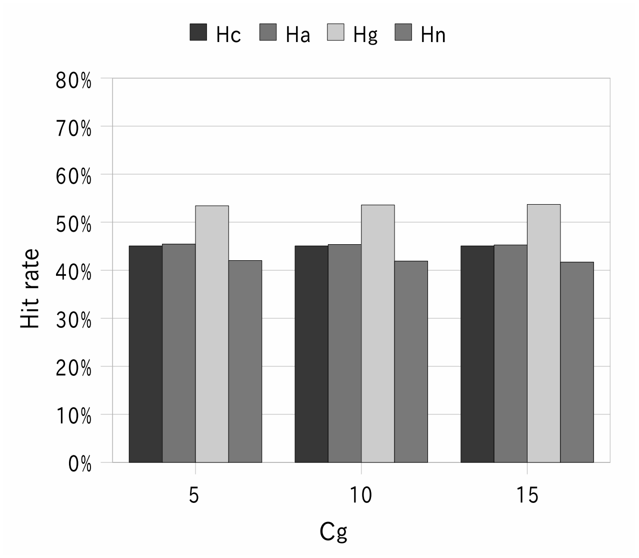

Figure 10. The number of nodes was 100, Ca$= 50$, and Gr$= 25\%$.

Fig. 7 and Hn in Fig. 8 are slightly larger than the respective corresponding hit rates.

4.4. Storage size for groups

Lastly, we measured hit rates with varying Cg for observing the influence of the storage size upon the system performance. Fig. 10 shows hit rates measured in cases where we set the number of nodes to 100, Ca$= 50$, and Gr$= 25\%$, and varied Cg as 5, 10, and 15. As the storage size for groups grows, Hg increases and Hn decreases very slightly, and as a result Ha decreases very slightly. The changes among them are very small by comparing with the differences of the values of Cg. In order to predict the appropriate storage size for groups, we intend to develop some cost model in future.

5. Conclusion

In this paper we proposed group participation support for decentralized P2P web caching systems where groups can be built with nodes. The support allows us with several nodes to participate in the caching system as a group, and to get better performance by means of group participation. The system thus can have more peers in the network. Some simulation results show that, no matter a node belongs a group or not, our system can give the node advantages in terms of the hit rate.

As we wrote before, we need to analyze and optimize the hit rate in terms of factors including the number of peers, the size of cache storage, group participation ratios, and the cache size for groups. We also need to consider other distributions of web object popularities like [19]. Also, we intend to develop a real P2P web caching system and evaluate it in the real world. In it, we will consider the size of web objects, similar to [20], and access latency occurred in the real system.

References

[1] L. Xiao, X. Zhang, and Z. Xu, "On reliable and scalable peer-to-peer web document sharing," in *Proceedings of International Parallel and Distributed Processing Symposium (IPDPS 2002)*. IEEE Computer Society, 2002, p. 0023b.

[2] K. Kim and D. Park, "Efficient and scalable client clustering for web proxy cache," *IEICE Trans. Inf. & Syst.*, vol. 86-D, no. 9, pp. 1577–1585, Sep. 2003.

[3] Z. Xu, Y. Hu, and L. Bhuyan, "Exploiting client cache: A scalable and efficient approach to build large web cache," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*. IEEE Computer Society, 2004, p. 55a.

[4] S. Iyer, A. Rowstron, and P. Druschel, "Squirrel: A decentralized peer-to-peer web cache," in *PODC '02: Proceedings of the Twenty-First Annual Symposium on Principles of Distributed Computing*. Monterey, California: ACM, 2002, pp. 213–222.

[5] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*. Springer, 2001, pp. 329–350.

[6] B. Sheng and F. B. Bastani, "Secure and reliable decentralized peer-to-peer web cache," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*. IEEE Computer Society, 2004, p. 54b.

[7] H. Cheng, Z. Gu, and J. Ma, "Intracache: An interest group-based P2P web caching system," in *Proceedings of 2007 IEEE International Parallel and Distributed Processing Symposium (IPDPS 2007)*. IEEE Computer Society, 2007, p. 419.

[8] H.-K. Chiang, H.-W. Chen, and F.-L. Kuo, "Locality support for mobile P2P network," in *IWCMC '07: Proceedings of the 2007 International Conference on Wireless Communications and Mobile Computing*. Honolulu, Hawaii, USA: ACM, 2007, pp. 517–522.

[9] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, 2003.

[10] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao, "Distributed object location in a dynamic network," in *SPAA '02: Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*. Winnipeg, Manitoba, Canada: ACM, 2002, pp. 41–52.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 161–172, 2001.

[12] H. V. Jagadish, B. C. Ooi, and Q. H. Vu, "Baton: a balanced tree structure for peer-to-peer networks," in *VLDB '05: Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005, pp. 661–672.

[13] M. Junginger and Y. Lee, "The multi-ring topology – high-performance group communication in peer-to-peer networks," in *P2P '02: Proceedings of the Second International Conference on Peer-to-Peer Computing*. IEEE Computer Society, 2002, p. 49.

[14] G. Xue, Y. Jiang, Y. You, and M. Li, "A topology-aware hierarchical structured overlay network based on locality sensitive hashing scheme," in *UPGRADE '07: Proceedings of the Second Workshop on Use of P2P, GRID and Agents for the Development of Content Networks*. Monterey, California, USA: ACM, 2007, pp. 3–8.

[15] A. Anglade, M. Tiemann, and F. Vignoli, "Complex-network theoretic clustering for identifying groups of similar listeners in P2P systems," in *RecSys '07: Proceedings of the 2007 ACM Conference on Recommender Systems*. Minneapolis, MN, USA: ACM, 2007, pp. 41–48.

[16] K. Shudo, Y. Tanaka, and S. Sekiguchi, "Overlay weaver: An overlay construction toolkit," *Computer Communications*, vol. 31, no. 2, pp. 402–412, 2008.

[17] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99)*, 1999, pp. 126–134.

[18] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy, "Organization-based analysis of web-object sharing and caching," in *USITS'99: Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*. Boulder, Colorado: USENIX Association, 1999, pp. 25–36.

[19] W. Rao, L. Chen, A. W.-C. Fu, and Y. Bu, "Optimal proactive caching in peer-to-peer network: Analysis and application," in *CIKM '07: Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*. Lisbon, Portugal: ACM, 2007, pp. 663–672.

[20] K. Kim and D. Park, "Efficient and tailored resource management for the P2P web caching," *IEICE Trans. Inf. & Syst.*, vol. 90-D, no. 1, pp. 48–57, Jan. 2007.