International Journal For Development of Computer Science & Technology (IJDCST), Vol-1, Issue-1, Jan-2013

An Adaptive File Replication Scheme in P2P System

Tummala Bindu Madhavi#1, G.Venkata Krishna#2

#1 Student, Dvr & Dr. Hs Mic College Of Technology, Kanchikacherla,Krishna(dt)

#2 Asst. professor, Dvr & Dr. Hs Mic College Of Technology, Kanchikacherla,Krishna(dt)

#1 bindhu.tummala@gmail.com, #2 krishna4maths@yahoo.com,

Abstract: File replication and File consistency maintenance are widely used techniques in peer-to-peer file sharing systems for high system performance and optimum utility of P2P networks. Despite significant interdependencies between them, these two issues are typically addressed separately. Existing file replication methods rigidly specify replica nodes, leading to underutilization of replicas, clumsy performance with unnecessary replicas and hence require extra consistency maintenance overhead. Most consistency maintenance methods updates, propagates messages operating without considering the issues involved in file replication dynamism, leading to inefficient file update and hence high possibility of outdated file response. This paper presents an integrated file Replication and consistency maintenance mechanism that integrates the two techniques in a cohesively harmonized manner. It achieves high efficiency in file replication and consistency maintenance at a significantly low cost. The mechanism's reliance on polling file owners still cannot guarantee that all file requesters receive up-to-date files, although its performance is better than other consistency maintenance algorithms. Instead of passively accepting file replicas and updates, each node avoids unnecessary file replications and determines update polling by adapting to time-varying file query and update rates. Simulation results demonstrate the effectiveness of proposed system in comparison with other earlier approaches. It reduces overhead and yields significant improvements on the efficiency of both file replication and consistency maintenance approaches.

I INTRODUCTION

A peer-to-peer (P2P) file sharing system consists of peer nodes (computers, devices, etc.) that want to share their resources and store files cooperatively to improve their own utilities and/or provide services to the users. For example, a P2P file sharing system can provide *document backup services* for important files. By storing multiple copies of a file in several nodes possibly at different locations, it can improve the *availability* of the file in case of fault, failure, or disaster. It can also provide load balancing and reduce access latency if the file is accessed by a large population of users.

The *file replication scheme* is a key component of any file sharing system. It determines the number of copies of the files to be stored in the system as well as in which nodes to store those file copies in order to achieve certain system objectives. The file copy numbers and locations can be determined either jointly or separately.

File replication is an effective method to deal with the problem of overload condition due to flash crowds or hot files It distributes load over replica nodes and improves file query efficiency by reducing server response latency and lookup path length (i.e., the number of hops in a lookup path). A higher effective file replication method produces higher replica hit rate. A replica hit rate occurs when a file request is resolved by a replica node than the file owner. File consistency maintenance is to maintain the consistency between a file and its replicas is indispensable to file replication. Thus file replication should reduce unnecessary replicas to reduce consistency maintenance overhead.

In a distributed file system, there are two essential aspects for a replica management policy to decide: the replication degree and the allocation of replicas on different nodes. Depending on the moment at which decisions are taken, there are two types of replication policies: static and dynamic. A static replication policy decides the replication degree and the allocation of replicas at the moment when the file enters the system, and this decision will not change during the lifetime of the file. In a dynamic replication policy, both the replica degree and the allocation of the replicas can change during the lifetime of the file according to the current situation.

Recently, numerous file replication methods have been proposed. But these methods either have low effectiveness on improving query efficiency or come at a cost of high overhead. These methods make it difficult to adjust the number of replicas to time-varying utilization of replicas and to ensure that all replicas are fully utilized. The number of replicas has a significant impact on the consistency maintenance. More replicas lead to high consistency maintenance overhead and vice versa. Without taking into account file replication dynamism, consistency maintenance generates unnecessary overhead and cannot help to guarantee the fidelity of replica consistency.

In this paper, we present a mechanism an ADAPTIVE FILE REPLICATION SCHEME that achieves high efficiency in file replication and consistency maintenance at a significantly lower cost. We Propose to use an update polling method that adapts to time-varying file queries and update rates instead of polling file owners at random intervals and yields significant performance gains. Proposed method avoids unnecessary file replications and updates by dynamically adapting to time varying file query and update rates. It improves replica utilization, file query efficiency, and consistency fidelity.

II RELATED WORK

The file replication methods copies files near file owners, file requesters or along a query path from a requester to a owner. PAST, CFS, and Backslash replicate each file on close nodes near the file's owner. In LAR and Gnutella, overloaded nodes replicate a file at requesters. In these methods, file owners rigidly determine replica nodes and nodes accept replicas. They are unable to keep track replica utilization to reduce underutilized replicas and ensure high utilization of existing replicas.

CUP is a protocol for performing Controlled Update Propagation in P2P networks. The propagation is conducted by building a CUP tree. A node received updates for metadata items only if the nodes has registered interest with its neighbor. However intermediate nodes along the path receive the updates they do not need it. Envoy is a two-layer P2P network where a structured overlay is build on top of an unstructured one. The purpose of using the twolayer architecture is to combine the advantage of each structure and create synergy. Structured overlay, on the other hand, guarantees every search to be completed in bounded steps, typically in logarithmic of the network size. Therefore, by combining the two structures, both popular and rare/distant objects can be effectively and efficiently located.

YAPPER combines both structured peer-to-peer networks and unstructured peer-to-peer networks to provide a scalable lookup service over an arbitrary topology. Both data keys and peers are hashed to different buckets or colors. Data is stored in the peer in the same color. Finally, all the peers in the same color will be checked. However, YAPPERS is designed for efficient partial lookup that only returns partial values of data. For a complete lookup, YAPPERS still needs to flood the request to all peers that are in the same color as the data.

BitTorrent is a centralized unstructured peertopeer network for file sharing. When a peer has received the complete file, it should stay in the system for other peers to download at least one copy of the file from it. Since BitTorrent uses a central server to store all the information BitTorrent is a centralized unstructured peertopeer network for file sharing. When a peer has received the complete file, it should stay in the system for other peers to download at least one copy of the file from it. Since BitTorrent uses a central server to store all the information.

Raunak et al. proposed polling method for web cache consistency. Its context is static, in which the proxies are always available. Thus, this method is not applicable for a P2P dynamic environment. Muthitacharoen et al. proposed lease algorithm for consistency. The lease is a commitment on the part of the server to notify the client of any modifications made to that file during the term of the lease. When the lease has expired, a client must request the server to transfer the new contents to itself. Rsync is a software application for UNIX systems which synchronizes files and directories from one location to another while minimizing data transfer using delta encoding when appropriate. An important feature of rsync not found in most similar protocols is that the mirroring takes place with only one transmission in each direction.

Along with the file replication methods, numerous file consistency methods have been proposed. Generally they are classified into two categories: structure based and message spreading based. The work in SCOPE constructs a tree for update propagation and in constructs a hierarchical structure with locality consideration. DUP propagate update along a routing path. In message spreading based methods, hybrid push/ poll algorithm flooding is replaced by rumor spreading to reduce communication overhead. In these methods updates are not guaranteed to be propagated to each replica and redundant message will generate high overhead. FreeNet replicates file on the path from the file requester to the target routes and it routes an update to other nodes based on key closeness.

III AN ADAPTIVE FILE REPLICATION SCHEME

Instead of passively accepting replicas and update messages, an adaptive file replication scheme integrates file replication and consistency maintenance by allowing each node to determine file replication and consistency maintenance based on file query rate and update rate. Proposed scheme aims to guarantee consistency maintenance at a low cost with file replication dynamism and uses adaptive polling to ensure timely update operation and avoids unnecessary updates.

A) File Replication

Ultimate objective of proposed scheme is to achieve high query efficiency and low file replication overhead. It dynamically adapts to time-varying file popularity and node interest, and adaptively determines replica nodes based on query traffic.

(i)File Replica nodes determination: Whenever a popular file like audio and video is accessed continuously, the server will be degraded and will give delayed response .In that situation placing a replica for that file is suitable.

Based on this idea, proposed system replicates a file in nodes that have been very interested in the file or query forwarding nodes that have been carrying more query traffic of the file. The former provides files to the frequent file requesters without query routing, increasing replica hit rate and the latter increases the probability that queries from different directions encounter the replica nodes.

(ii) *File Replica creation*: Creation of a replica is based on file query rate. A replica is created when the requesters request continuously a file. If the query rate is less to that replica., then the replica node is deleted. Proposed scheme sets a threshold for query rate denoted by *Tq* based on normal query initiating and passing rates in the system. We define query rate of a file 'f' denoted by *qf* as the number of queries initiated by a requester or forwarded by a node during unit time period.

If $qf > Tq$ and it has enough capacity for a file replica, it piggybacks a file replication request and its qf into a file query when initiating or forwarding a file request for this file.

If a nodes $qf > Tq$, it is regarded as frequent requester or traffic hub for a file 'f'.

After the file destination receives the query, if it is overloaded then it checks the file query for additional file requesters. If so then it sends the file to the file replication requesters in addition to the query initiator.

(iii) File Replica adaptation: A file might not be accessed frequently .so the replica node should frequently update their query passing rate, so that underutilized replicas can be removed. In proposed scheme, at periodic intervals each replica calculates their query passing rate or query initiating rate of a file. If the rates are below their thresholds, the node removes the replica. By doing this, if a file is no longer requested frequently, there will be no file replica for it. By implementing this, there is no waste of overhead for unnecessary file consistency maintenance.

(B) Replica Consistency Maintenance

In proposed scheme, poll-based consistency maintenance, each replica node polls its file owner or another node to validate whether its replica is the up-to-date file, and updates its replica accordingly.

For poll-based consistency maintenance, different polling mechanisms are specified while implementing suitable polling technique is selected:

Poll Every Time: In this policy, the peer polls the owner every time a query or a download request for the file is received. This is a lazy polling policy since a poll is triggered only when necessary.

Periodic Polls: An alternate approach is to poll the owner periodically to check if the file has been updated. The periodicity of the polls is statically determined by the owner and depends on how frequently the file is expected to be modified.

Adaptive Polls: Rather than determining the poll frequency statically, a third approach is to dynamically vary the polling frequency based on the update rate for the file. A peer can observe that rate of updates to a file and poll more frequently when the file is being modified frequently and less frequently when it is not.

(i) Polling time determination: Consider file maximum update rate is $1/\Delta t$, which means it update every Δt time units. In proposed scheme, a replica node adjusts its polling frequency so that it polls at approximately the same frequency of file change.

Proposed scheme associates a time-to-update(TTU) value with each replica. A time-to-update (TTU) value is associated with each file. The TTU denotes the next time instant the peer must poll the owner, and thus, determines the polling frequency. The TTU value is varied dynamically based on the results of each poll message. The TTU value is increased by an additive amount if the file doesn't change between successive polls. In the event the file is updated since the last poll, the TTU value is reduced by a multiplicative factor.

If the file is modified since the last poll then the TTU value is

$$TTU = TTUold + A$$

Where A, $A > 1$ is the multiplicative decrease constant.

If the file doesn't change between successive polls:

$$TTU = TTUold + B$$

Where B, $B>0$, is an additive constant.

To adapting the TTU to the update rate, we take into account the number of active neighbors of a peer when computing the TTU. In general, a peer should poll more frequently when the network sees frequent joins and leaves, since frequent changes to the overlay topology increases the probability of missing an invalidate message. Suppose that a peer has N_{conn} active connections to its neighbors and let $N_{angconn}$ denote the average connectivity of a peer in the network. The TTU is chosen more aggressively when the number of neighbors drops below average and is made larger when a peer is well-connected and has more neighbors than the average peer.

$$TTU= TTU +(1+ (N_{conn} - N_{angconn})/ N_{angconn}) * \alpha$$

Where α is a constant. The TTU is decreased if thepeer has a small number of neighbors and increased otherwise. This TTU value is constrained by the maximum and minimum allowable TTU values $TTU_{max}$and TTU_{min}. $TTU_{max}$and TTU_{min} are upper and lower bounds of TTU values.

(ii)Reduction in polls: Along with file change rate, file query rate also a major role in consistency maintenance. When a file changes frequently, if a replica node does not receive queries for the file, or hardly queries for the file during a time period, it is an overhead waste to poll the file's owner for validation during the time period.

Here, we use *TTUquery* and *TTUpoll* to denote the next time instant of corresponding operation of a file. When *TTU > Tquery*, that is, the file is queried at a higher rate than change rate, then the file should be updated timely based on *TTU*. On the other hand, when *TTU ≤ TTUquery*, that is, when the file change rate is higher than the file query rate, there is no need to update the replica at the rate of file change rate.

IV PERFORMANCE

For optimum utility of P2P networks many file replication and file consistency maintenance techniques were used. Many techniques have high computation overhead, implementation costs and are yielding only moderate results. Uses an integrated mechanism for both file replication and file consistency maintenance. In this paper, we use an update polling method that adapts to time-varying file queries and update rates instead of polling file owners at random intervals. The mechanism relies on polling file owners and then implementing replication and consistency maintenance procedures. By implementing proposed method, performance can be improved and reduces computation overhead, implementation costs and is yielding optimum results. Considering the ever changing dynamics of P2P systems this adaptive polling method yields significant performance gains.

V CONCLUSION

Many file replication methods replicate files close to file owners, file requesters, or query path to release the owners load, and to improve the file query efficiency and performance. File replication needs consistency maintenance to keep the consistency between a file and its replicas, and on the other hand, the overhead of consistency maintenance is determined by the number of replicas. In this paper, we propose an Adaptive File Replication Scheme that achieves high efficiency at a significantly low cost. We propose to use an update polling method that adapts to time-varying file queries and update rates instead of polling file owners at random intervals. Considering the ever changing dynamics of P2P systems this adaptive method yields significant performance gains. Instead of passively accepting replicas and updates, nodes autonomously determine the need for file replication and validation based on file query rate and update rate. It guarantees high utilization of file replicas, query efficiency and consistency. Implementing proposed scheme reduces redundant file replicas, consistency maintenance overhead by polling approach.

VI REFERENCES

[1] A Rowstron and P. Druschel, "Storage Management and Caching in PAST, a Large-Scale, Persistent Peer to- Peer Storage Utility," Proc. Symp. Operating Systems Principles (SOSP), 2001.

[2] V. Gopalakrishnan et al., "Adaptive Replication in Peer to-Peer Systems," Proc. Int'l Conf. Distributed Computing Systems (ICDCS), 2004

[3] M. Roussopoulos and M. Baker. CUP: Controlled Update Propagation in Peer to Peer Networks. In *Proc. Of USENIX*, 2003.

[4] R. Cox, A. Muthitacharoen, and R.T. Morris, "Serving DNS Using a Peer-to-Peer Lookup Service," Proc. Int'l Workshop Peer to-Peer Systems (IPTPS), 2002.

[5] H. Shen, "EAD: An Efficient and Adaptive Decentralized File Replication Algorithm in P2P File
Sharing Systems," Proc. Eighth Int'l Conf. Peer-to-Peer Computing (P2P '08), 2008.

[6] X. Chen, S. Ren, H. Wang, and X. Zhang. SCOPE: scalable consistency maintenance in structured P2P systems. In *Proc. of INFOCOM*, 2005.

[7] L. Yin and G. Cao. DUP: Dynamic-tree Based Update Propagation in Peer-to-Peer Networks. In *Proc. Of ICDE*, 2005.

[8] Z. Li, G. Xie, and Z. Li. Locality-Aware Consistency Maintenance for Heterogeneous P2P Systems. In *Proc.of IPDPS*, 2007.

[9] A. Datta, M. Hauswirth, and K. Aberer. Updates in Highly Unreliable, Replicated Peer-to-Peer Systems. In *Proc. of ICDCS*, 2003.

[10] I. Clarke, O. Sandberg, and et al. Freenet: A Distributed Anonymous Information Storage and Retrieval System.