International Journal of Electronics and Electrical Engineering

Volume 1 | Issue 1 Article 3

July 2012

EARM: An Efficient and Adaptive File Replication with Consistency Maintenance in P2P Systems

K. V. K. Chaitanya
Department of CSE, JNTUACE, Anantapur, A.P, chaitanya.kvk604@gmail.com

S. Vasundra
1 ,Department of CSE , JNTUACE, Anantapur, A.P, vasundaras@rediffmail.com

Recommended Citation

Chaitanya, K. V. K. and Vasundra, S. (2012) "EARM: An Efficient and Adaptive File Replication with Consistency Maintenance in P2P Systems," *International Journal of Electronics and Electrical Engineering*: Vol. 1 : Iss. 1 , Article 3.
DOI: 10.47893/IJEEE.2012.1002
Available at: https://www.interscience.in/ijeee/vol1/iss1/3

EARM: An Efficient and Adaptive File Replication with Consistency Maintenance in P2P Systems.

[1]K.V.K.Chaitanya, [2] Smt. S.Vasundra, M,Tech., (Ph.D),
[1]M.Tech (Computer Science), [2]Associate Professor,
Department of CSE,
JNTUACE, Anantapur, A.P.
E-mail: chaitanya.kvk604@gmail.com , vasundaras@rediffmail.com

***Abstract:** In p2p systems, file replication and replica consistency maintenance are most widely used techniques for better system performance. Most of the file replication methods replicates file in all nodes or at two ends in a client-server query path or close to the server, leading to low replica utilization, produces unnecessary replicas and hence extra consistency maintenance overhead. Most of the consistency maintenance methods depends on either message spreading or structure based for update message propagation without considering file replication dynamism, leading to inefficient file update and outdated file response. These paper presents an Efficient and Adaptive file Replication and consistency Maintenance (EARM) that combines file replication and consistency maintenance mechanism that achieves higher query efficiency in file replication and consistency maintenance at a low cost. Instead of accepting passively file replicas and updates, each node determines file replication and update polling by adapting to time-varying file query and update rates. Simulation results demonstrate the effectiveness of EARM in comparison with other approaches.*

***Key words:** File Replication, consistency maintenance, peer-to-peer, distributed hash table.*

I. Introduction

Peer-to-peer (P2P) systems have become a popular way for file sharing and distribution where a file requester's query is forwarded to a file provider in a distributed manner. A recent large-scale characterization of HTTP traffic [1] has shown that more than 75 percent of Internet traffic is generated by P2P file sharing applications. The study shows that the access to these files is highly repetitive and skewed toward the most popular ones. In such cases, if server receives many requests at a time, it becomes overloaded and cannot respond to queries quickly. 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. Replica hit rate denotes the percentage of the number of file queries that are resolved by replica nodes among total queries. 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.Recently, numerous file replication methods have been proposed. The methods can be generally classified into three categories namely ServerSide, ClientSide, and Path. ServerSide replicates a file close to the file owner [2], [3]; ClientSide replicates a file close to or at a file requester [4], [5]; and Path replicates on the nodes along the query path from a requester to a file owner [6], [7]. However, most of 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. In EAD[8], traffic hubs and frequently requesters are chosen as replica nodes. These nodes periodically compute their query load to create replicas and remove underutilized replicas. EARM is developed by leveraging EAD. It shares similarity with EAD in file replication strategies. The number of replicas has a significant impact on the consistency maintenance. More replicas lead to high consistency maintenance overhead and vice versa.

Most consistency maintenance methods update files by relying on a structure [9], [10],[11] or message spreading [12],[13]. Though these methods generally can be applied to all file replication methods, they cannot be exploited to their full potential without considering time varying and dynamic replica nodes. Message spreading generated high overhead due to redundant messages, and cannot guarantee that every replica node receives an update. Structure-based pushing methods reduce the overhead but cannot guarantee timely consistency in churn. Therefore, without taking into account file replication dynamism, consistency maintenance generates unnecessary overhead and cannot help to guarantee the fidelity of replica consistency.

This paper presents a mechanism an Efficient and Adaptive file Replication and consistency Maintenance (EARM) that achieves high efficiency in file replication and consistency maintenance at a significantly lower cost. It combines file replication and consistency maintenance in a harmonized and coordinated manner. Instead of creating replicas on all nodes or two ends on a client-server path, it chooses query traffic hubs (i.e., query traffic conjunction nodes) as replica nodes to ensure high replica hit rate. It replicates highly queried files and polls at a high frequency for frequently updated and queried files. EARM 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

File Replication in p2p systems aims to release load in hot spots and to decrease file query latency. Numerous file replication methods are proposed and these are classified into three categories: ServerSide, ClientSide, and Path.

In ServerSide replication includes CFS [2], PAST [3] which replicates the file at the file owner. PAST is an Internet-based global P2P storage utility with a storage management and caching system. It replicates files at the nodes whose nodeIDs match closely with the file owner's nodeIDs. PAST uses file caching along the lookup path to minimize query latency and balance query load. Cooperative File System (CFS) [2] is a P2P read-only storage system for file storage and retrieval. CFS is built on Chord and replicates blocks of a file on nodes immediately after the block's owner on the Chord ring.

In ClientSide category, Gnutella [4], LAR [5] replicates file close to the file owner. Gnutella [4] replicates files in overloaded nodes at the file requesters. LAR [5] is a lightweight, adaptive, and system-neutral replication protocol. It specifies the overloaded degree of a server that a file should be replicated, and replicates a file to a client. In addition to replicating a file at the requester, LAR also replicates file location hints along the lookup path

In the Path category, CUP [6] 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.

Along with the file replication methods, numerous file consistency methods have been proposed. Generally they are classified into two categories: structure based [9], [10], [11] and message spreading based [12], The work in SCOPE [9] constructs a tree for update propagation and in [11] constructs a hierarchical structure with locality consideration. DUP [10] propagate update along a routing path. In message spreading based methods, hybrid push/poll algorithm [12] 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 [13] 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. EARM: Efficient and Adaptive File Replication and Consistency Maintenance Mechanism

Instead of passively accepting replicas and update messages, EARM 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. EARM replicates file in frequently visited nodes to guaranteed high replica utilization, and remove under-utilized replicas and overhead of consistency maintenance. EARM aims to guarantee consistency maintenance at a low cost with file replication dynamism. EARM uses adaptive polling to ensure timely update operation and avoids unnecessary updates.

A. File Replication:

EARM ultimate objective 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. EARM addresses two main issues to achieve its goals: (1) where to replicate files so that the file query can be significantly expedited and the replicas can be fully utilized? (2) how to remove under-utilized file replicas so that the overhead for consistency maintenance is minimized?

1). File Replica nodes determination: In Structured p2p systems, the query load is distributed in an imbalance manner. Because of these some nodes carry more query traffic while other carry less. Therefore, frequent requesters of a file and traffic junction nodes in query paths should be the ideal file replica nodes for high utilization of file replicas. Based on this, EARM 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.

2) File Replica creation: We define query rate of a file 'f' denoted by q_f as the number of queries initiated by a requester or forwarded by a node during unit time period. EARM sets a threshold for query rate denoted by T_q based on normal query initiating and passing rates in the system. If a nodes $q_f > T_q$, it is regarded as frequent requester or traffic hub for a file 'f'. If $q_f > T_q$ and it has enough capacity for a file replica, it piggybacks a file replication request and its q_f into a file query when initiating or forwarding a file request for this file. After the file destination receives the query, if it is overloaded, it checks if the file query has additional file requesters. If so it sends the file to the file replication requesters in addition to the query initiator.

3). File Replica adaptation: As file popularity is non uniform and time varying and node interest varies over time, some file replicas become unnecessary when there are few queries for files. In previous methods, a file server maintains information of its replica nodes to manage the replicas and spreads information about new replica sets. EARM makes replica adjustments in a decentralized manner. EARM lets each replica node periodically calculate 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. The adaptation to these query initiating and passing rates will ensure that all replicas are worthwhile and there is no waste of overhead for unnecessary file consistency maintenance.

B. Replica Consistency Maintenance:

In EARM 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. EARM addresses three main issues in consistency maintenance. (1) how to determine the frequency that a replica node probe a file owner in order to guarantee timely file update? (2) How to reduce the number of polling operations to save cost?

1) *Polling time determination:* Consider file maximum update rate is $1/\Delta t$, which means it update every Δt time units. Therefore, a file replica node can ensure that a replica will not be outdated by more than Δt seconds by polling the owner every Δt seconds. Since the rate of file change varies over time as hot file become cold and vice versa, a replica node should be able to adapt to the changes dynamically. In EARM, a replica node adjusts its polling frequency so that it polls at approximately the same frequency of file change using a *linear increase multiplicative decrease* algorithm.

EARM associates a time-to-update(TTU) value with each replica. The TTU denotes the next time instant a node should poll the file owner to keep its replica updated. The TTU value is varied dynamically based on the results of each polling. The value is increased by an additive amount if the file doesn't change between successive polls:

$$TTU = TTU_{old} + \alpha \qquad (1)$$

Where α, α>0, is an additive constant. In the event if the file is updated since the last poll, the TTU value is reduced by a multiplicative factor:

$$TTU = TTU_{old} + \beta \qquad (2)$$

Where β, β > 1 is the multiplicative decrease constant. The algorithm takes as input two parameters TTU_{min} and TTU_{max} which represents lower and upper bounds on TTU value. Values that fall outside these bounds are set to

$$TTU = max\,(TTUmin,\ min\,(TTUmax,\ TTU)) \qquad (3)$$

These bounds are required to ensure that TTU values are neither too large nor too small. The TTU value set to Δt , the minimum interval between polls necessary to maintain consistency.

2) *Reduction in polls:* Along with file change rate, file query rate also a major role in consistency maintenance. Even 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. EARM combines file query rate into consideration for poll frequency determination. 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. Therefore, the polling rate can be equal to file query rate in this case.

IV. Simulation Results

TABLE I
SIMULATION SETUP

operating system	Red hat Linux 9
Simulator	NS2(Network Simulator2)
Topology	Wireless topology
Number of nodes	16
Simulation time	40sec
Area of the Network	1000m*1500m

Network Simulator(NS2) is used for simulating the existing and proposed systems. NS2 is an IEEE standardized simulator for simulating Networks. Table I lists the simulation parameters their default values.

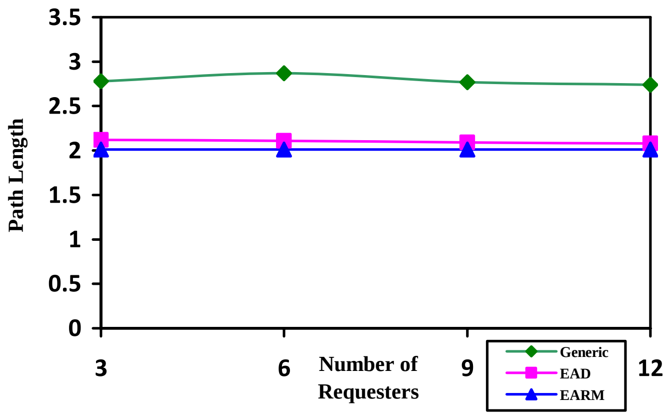

Figure 1:Path length of Generic, EAD and EARM

Consider different number of file requesters and compare the results for Generic(without file replication), EAD and EARM with respect to the path length and consistency maintenance. The Figure 1. graph contains three file replication protocols path length. In the graph, Generic is represented with green, EAD is represented with pink, EARM is represented with blue. The simulation results in the Figure 1. shows that path length from requestor to file owner in EARM is less when compared with EAD and Generic.

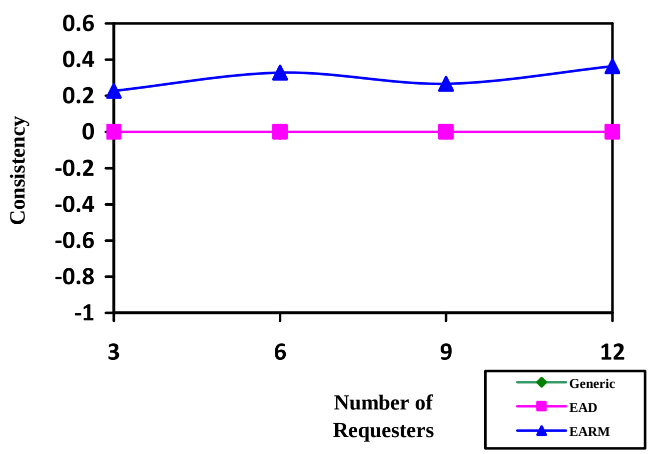

Figure 2: Consistency Maintenance of Generic, EAD and EARM

The Figure 2. graph contain three replication protocols consistency maintenance. The simulation results show that EARM provides better file replication with consistency maintenance where as EAD and Generic provides no consistency. Here consistency is measured in terms of update messages detected by the server.

IV. 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. But all of these methods have disadvantages that will lead to decrease in the file query efficiency, replica hit rate and increase in overload. In spite of the effort to develop file replication and file consistency maintenance in p2p systems, there has been very little research devoted to tackle both challenges simultaneously. 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. Connecting these two components will increase the system performance.

This paper proposes an EARM that achieves high efficiency at a significantly low cost. It chooses query traffic hubs and frequent requesters as replica nodes to guarantee high utilization of replicas and high query efficiency. 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. At the same time, EARM reduces redundant file replicas, consistency maintenance overhead by polling approach. Simulation results demonstrate the effectiveness of EARM in comparison with other file replication approaches.

References

[1] S. Saroiu, P. Gummadi, and S. Gribble, “A Measurement Study of Peer-to-Peer File Sharing Systems,” Proc. Conf. Multimedia Computing and Networking (MMCN), 2002.

[2] F. Dabek et al., “Wide Area Cooperative Storage with CFS,” Proc. Symp. Operating Systems Principles (SOSP), 2001.

[3] 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.

[4] Gnutella, http://www.gnutella.com, 2008.

[5] V. Gopalakrishnan et al., “Adaptive Replication in Peer-to-Peer Systems,” Proc. Int’l Conf. Distributed Computing Systems (ICDCS), 2004

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

[7] 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.

[8] 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.

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

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

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

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

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