ENHANCED EFFICIENT INFORMATION SHARING IN CLOUD ENVIRONMENT USING EERA

S.Sajithabanu, Research Scholar, Department of Computer Applications, NIT, Trichy. 
sajithabanu.phd.2013@gmail.com

Dr.S.R.Balasundaram, Associate Professor, Department of Computer Applications, NIT, Trichy. 
blsundar@nitt.edu

Abstract - Nowadays, Information storage and access through cloud network is becoming an economical alternative for delivery of traditional content. Mostly the content delivery is based on the moderate-size content providers. In Hybrid cloud network the services are be a subjected to momentous popularity increase due to its strategies and its performance of resources access. For optimization problem in cloud storage we have designed an EERA (Enhanced Efficient Replication Algorithm) for our optimization problem. The proposed work focused on reducing the bandwidth usage, traffic, delay access and latency also the performance cost and access time for storing and content delivery. Based on content provider the performance and results evaluation is take place in cloud environment. We implemented an effective and robust approximation algorithm in the proposed work and evaluated the complexity of the algorithm. The performance of the work provides a light-weight process, less complexity for time and space as well as cost, better accuracy of access of data and finally efficient distributed access in hybrid cloud environment.

Keywords: Cloud storage, Content delivery, Replication.

1 INTRODUCTION

Content Delivery Networks (CDNs) have progressed to overwhelm the intrinsic limitations of the Internet in terms of user alleged Quality of Service (QoS) when retrieving web content. A Content Delivery Network replicates video content from the origin server to proxy servers, disseminated over the world, so as to distribute content to end-users in a trustworthy and well-timed manner from nearby optimal proxies. It integrates development of high-end computing technologies with high-performance networking structure and dispersed replica management techniques.

Content Delivery Networks (CDNs) [1][2] provide services that improve network performance by increasing bandwidth, improving availability and retaining correctness through content replication.

They provide reckless and reliable applications and services by dispensing content to cache or edge servers located close to users [1]. A CDN has some mixture of content-distribution, request-routing, scattering and accounting infrastructure. The content-distribution structure consists of a group of edge servers (also called surrogates) that distribute replicas of content to end-users.

2 COST OPTIMIZATION

As a traditional content distribution network consists of a central origin server and multiple proxy servers, a content distribution network over cloud storage is comprised of an origin server and multiple proxy servers on a cloud network. The proxy servers are connected with the origin server and/or other proxy servers, as illustrated in below figure. In order to shorten the latency experienced by the final content users, some objects are replicated on a proxy server, also called a replica, in advance near the users. Moreover, in order to improve the content users’ experience, proxy servers may cooperate with each other so that they add objects from others rather than from the origin server. The proxy servers that are connected by a
direct overlay link can communicate with each other, called neighbours.

3 THE PUSH–PULL MODEL

Content users in area i will directly request and download object from the proxy server i. For simplicity of our discussion, we denote a proxy as if there is only one proxy server per one area. In practice, an origin server may be a server farm, and more than one proxy server may be in charge of one area. In such a case, the sum of proxy servers’ storage space for area i can be used as $ci$. Notice that cloud storage requires multiple replications based on predefined replication factor. The available storage space will be the actual storage size divided by the replication factor. Here, we assume that the same replication factor will be applied for all the contents, and use $ci$ as the available storage space for distinct contents. We point out that we focus on the performance within a distribution cloud network and do not compute the latency and traffic costs between a proxy server and the final content users. In fact, those costs will not be impacted by the content distribution strategies, since each content user is directly served by a fixed proxy server according to the users’ location. Therefore, our problem is to minimize the total cost, which includes the latency cost and the traffic cost between cloud servers, to satisfy all the requests from content users. Satisfying content users’ requests, two typical schemes can be considered for content distribution, a push-based scheme and a pull-based scheme. In a push-based scheme, objects are replicated into proxy servers prior to requests until a proxy server’s storage limitation is reached. This distribution procedure is referred to as a push.

4 CONTENT DELIVERY MODEL

When requests arise for those pushed objects, a proxy server will directly serve the content users without involving other cloud servers. On the other hand, no object is replicated into the proxy servers in advance under a pull-based scheme. When proxy servers receive queries for the un-pushed objects, they will forward the query and download the requested objects from the nearest source that could be either the origin server or another cooperative proxy server. This procedure is specified as a pull. By disseminating objects in advance, a push-based scheme shortens the retrieval latency of object j; while a pull-based scheme reduces traffic volume by eliminating object downloads that would never be requested. In order to optimize both latency cost and traffic cost, we consider a content delivery scheme that is the combination of a push-based scheme and a pull-based scheme. This push–pull scheme is expected to properly determine which objects should be pushed and which should be pulled, so that both the push-based scheme and pull-based scheme are used to their best advantages.

5 RELATED WORK

A hierarchical framework is proposed and evaluated toward an efficient and scalable solution of content distribution over a multiprovider networked cloud environment [11]. A clustering data placement strategy that can automatically allocate application data among data centres based on [12] hybrid architecture for cost-effective streaming media distribution. The architecture combines two complementary technologies: CDN and P2P [13]. A link-level measurement on network provider platforms as a basis for network-wide traffic engineering, including load balancing for current trends and prognosis. The main goal, namely to shorten end-to-end paths and delays and reducing the load in the backbone and on expensive global interconnection links and reduce energy consumption[15]. Measures the performance of the current Akamai platform [16] and considers a key architectural question faced by both CDN designers and their prospective customers whether the co-location approach to CDN platforms adopted by Akamai, which efforts to deploy servers in several Internet locations, brings intrinsic performance reimbursements over a more consolidated data
centre approach pursued by other influential CDNs such as Limelight[16]. To file transfer applications, heuristic distributed algorithm may find application in streaming services [19]. A Scaling decentralized content-based publish/subscribe (CBPS) networks proposed for large-scale content distribution [20]. The authors proposed a flexible agent-based modelling and simulation framework for designing and evaluating CDN architectures. It can be used to evaluate already existing solutions as well as to design and analyse novel solutions [21]. The authors proposed a simple structural approach for predicting Internet path latencies that out performs black box techniques. Latency predictions are based on an underlying path prediction model that can predict PoP-level paths with high accuracy [22]. The authors proposed a latency-sensitive content distribution mechanism, Space4time, for a real world system. They proposed a novel content distribution and request routing solution, Space4time (space for time). Based on blocking probability, Space4time effectively exploits the storage and network capacity for latency-sensitive applications [23].

### Table 1: Related Works

<table>
<thead>
<tr>
<th>S.NO</th>
<th>AUTHOR</th>
<th>JOURNAL</th>
<th>YEAR</th>
<th>WORK DONE</th>
<th>LIMITATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Mohamed Diallo &amp; at el</td>
<td>Elsevier</td>
<td>2012</td>
<td>Developed the publish/subscribe communication paradigm for content-based information retrieval which is used for improving communication efficiency.</td>
<td>Here the duplicate dropping heuristic does not thoroughly solve the problem addressed.</td>
</tr>
<tr>
<td>2.</td>
<td>Xinjie Guan &amp; at el</td>
<td>Elsevier</td>
<td>2013</td>
<td>Developed an approximation algorithm named Traffic-Latency- Minimization (TLM) for push-pull optimization problem which is used for content delivery from cloud storage.</td>
<td>DTLM algorithm may not find the most proper replica location for every object.</td>
</tr>
<tr>
<td>3.</td>
<td>Shijia. Yao, &amp; at el</td>
<td>JTAIT</td>
<td>2014</td>
<td>Developed two offline replica placement algorithms for cloud-based storage which is used to solved the load imbalance problems.</td>
<td>Proposed algorithms may not produce better results.</td>
</tr>
<tr>
<td>4.</td>
<td>J.Sun, S.Gao &amp; at el</td>
<td>JNW</td>
<td>2011</td>
<td>Developed an efficient algorithm (CPM) for allocate replicas of files to minimize the total cost. It includes three parts: replication algorithm pre-process, constraint P-median model and algorithm of solving constraint P-median problems which were solved by iteration methods. This CPM algorithm performs better than random algorithm with less total cost.</td>
<td>Limited Topology.</td>
</tr>
<tr>
<td>5.</td>
<td>Sharrukh Zaman &amp; at el</td>
<td>IEEE</td>
<td>2012</td>
<td>Developed a distributed approximation algorithm. The main focus is to improve the efficiency of popular object replication within a distributed replication group.</td>
<td>Degradation of system performance, not provide individual system performance, &amp; number of replicated copies.</td>
</tr>
<tr>
<td>6.</td>
<td>Yan Chen, &amp; at el</td>
<td>Springer</td>
<td>2002</td>
<td>A dynamic web content distribution system (d-tree) built on top of a peer-to-peer overlay network.</td>
<td>Limited local network topology knowledge.</td>
</tr>
<tr>
<td>No.</td>
<td>Author(s)</td>
<td>Conference/Journal</td>
<td>Year</td>
<td>Title</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-----------</td>
<td>--------------------</td>
<td>------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>8.</td>
<td>S. Zaman &amp; D. Grosu</td>
<td>IEEE</td>
<td>2011</td>
<td>Developed Distributed Algorithm for the Replica Placement Problem which is used to Minimizes Response Time, Cost &amp; Increase Availability.</td>
<td>Not provide individual system performance, &amp; number of copies</td>
</tr>
<tr>
<td>9.</td>
<td>J. Sun, &amp; al.</td>
<td>IEEC</td>
<td>2010</td>
<td>Developed Replica Placement Algorithm in Content Distribution Network which is used to minimize total Cost.</td>
<td>Can't adapt dynamic user request</td>
</tr>
<tr>
<td>10.</td>
<td>S. Zaman &amp; D. Grosu.</td>
<td>IEEE</td>
<td>2009</td>
<td>Developed Distributed Algorithm for Web Content Replication which is used to increase Response Time &amp; Cost, Availability &amp; Performance.</td>
<td>Not provide individual system performance, &amp; number of replicated copies</td>
</tr>
<tr>
<td>11.</td>
<td>S. Ayyasamy &amp; S. Natarajan</td>
<td>IJCSE</td>
<td>2009</td>
<td>Developed Intelligent Replica placement algorithm which is used to reducing both the access latency and Network traffic and increase the system performance.</td>
<td>High availability, Redundancy, Low Availability, Bottleneck Problem,</td>
</tr>
<tr>
<td>12.</td>
<td>N. Laoutaris &amp; al.</td>
<td>IEEE</td>
<td>2006</td>
<td>Developed a Two-Step Local Search (TSLS (k)) algorithm which employees Bloom Filters to distribute object, selected for replication in demanded patterns. It reduces bandwidth, access cost.</td>
<td>Does not consider storage capacity, too many rounds, additional protocols &amp; mechanisms.</td>
</tr>
<tr>
<td>14.</td>
<td>Alain Roy &amp; al.</td>
<td>IEEE</td>
<td>2010</td>
<td>Developed ERMS, an elastic replication management system for HDFS. ERMS provides an active/standby storage model for HDFS. Enhance the reliability and availability of data.</td>
<td>Does not detect and predict the real-time data types</td>
</tr>
<tr>
<td>15.</td>
<td>A. Rajalakshmi &amp; al.</td>
<td>IJIRSET</td>
<td>2014</td>
<td>Developed Dynamic Data Replica Selection and Placement in Hybrid Cloud and improved the overall data access performance and bandwidth utilization.</td>
<td>Does not extend replica placement in geographical locations</td>
</tr>
</tbody>
</table>
6 PROPOSED SYSTEM

The users upload the data to the origin server. The same data would be replicated to the many proxy servers to increase the response time. The two mechanisms used here. The push and pull mechanisms used for the clients to retrieve the data fast and efficiently. The latency in area a to object b corresponds with the distance $d_{ab}$ between proxy server a and its nearest replica of b. The latency from a proxy server to different users may vary depending on a user's access network and the path. In this paper, we point on the latency from the location of a requested object to the proxy server that directly serves a final client, and omits the latency from this proxy server to the final client.
Table 2: Notation

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>The number of proxy server</td>
</tr>
<tr>
<td>q</td>
<td>The number of video objects</td>
</tr>
<tr>
<td>a</td>
<td>Index of proxy servers also indicates the service area for server a, a=1,2,..p</td>
</tr>
<tr>
<td>c_a</td>
<td>The maximum storage space on proxy server a</td>
</tr>
<tr>
<td>b</td>
<td>Index of video objects, b =1,2,..q</td>
</tr>
<tr>
<td>d_{ab}</td>
<td>The distance from proxy server a to its nearest replica of video object b</td>
</tr>
<tr>
<td>\lambda_a</td>
<td>The total no. of request in area a.</td>
</tr>
<tr>
<td>P_{ab}</td>
<td>Probability that video object b will be queried in area a</td>
</tr>
<tr>
<td>S_b</td>
<td>The size of video object b</td>
</tr>
<tr>
<td>x_{ab}</td>
<td>Decision variable for pushing video object b to proxy server a</td>
</tr>
<tr>
<td>\alpha</td>
<td>Coefficient to balance the storage cost, content update cost, latency cost and bandwidth cost</td>
</tr>
<tr>
<td>k_{ab}</td>
<td>Cost for pulling video object b in area a</td>
</tr>
<tr>
<td>\Delta_{pq}</td>
<td>Total cost for a pull strategy with p proxy servers and q video objects</td>
</tr>
<tr>
<td>f_{ab}</td>
<td>Cost saved by pushing video object b to proxy server a</td>
</tr>
<tr>
<td>F_{pq}</td>
<td>Total cost for push pull strategy with p proxy servers and q video objects including latency and data traffic cost</td>
</tr>
</tbody>
</table>

Our problem is to minimize the total cost that includes the storage cost, content update cost, latency cost and the traffic cost and bandwidth cost between cloud servers to satisfy all the requests from content users. The latency in area a to obtain video object b corresponds with the distance d_{ab} between the proxy server at its nearest replica of b. The latency from a proxy server to different users may vary depending on a user’s access network and the path. The start-up time and frame rate of streaming video calculated.

When a request is raised for video content object b in area a if b has already pushed in to the proxy server a, proxy server a can respond immediately without involving any extra latency. However if video object b has not been replicated on proxy server a in advance, proxy server a needs to pull video object b from either the origin server or another cooperative proxy server that results in an additional latency cost d_{ab} for each request for video object b in area a.

Assume that the total number of requests for video object b in area a is the product of \lambda_a, which is the total number of requests in area a and P_{ab}, which is the probability that video object b will be queried in area a then the expectation of the total latency cost in area a is d_{ab}P_{ab}\lambda_a for pulled video object b.

\[ X_{ab} = \begin{cases} 1 & \text{video object b is pushed on proxy server a} \\ 0 & \text{video object b is not pushed on proxy server a} \end{cases} \]

Now the expectation of latency in a cloud based CDNs using the push- pull mechanism can be represented as

\[ \sum_{a=1}^{p} \sum_{b=1}^{q} (1 - X_{ab}) d_{ab} P_{ab} \lambda_a \] (2)

And the total traffic volume for this cloud based CDN can be formulated as

\[ \sum_{a=1}^{p} \sum_{b=1}^{q} (X_{ab} S_b + (1 - X_{ab}) S_b P_{ab} \lambda_a) \] (3)

And the average bandwidth for this cloud based CDN can be formulated as

\[ \sum_{a=1}^{p} \sum_{b=1}^{q} (S_b + Bf(t)) S_b P_{ab} \lambda_a \] (4)

We used a coefficient \alpha to balance the influence of latency and traffic costs and average bandwidth to satisfy the performance requirements.

Then our main objective is to minimize the total cost of latency and bandwidth and traffic and storage cost and content update cost for this network when using the Push-Pull mechanism under storage constraints.

\[ \sum_{a=1}^{p} \sum_{b=1}^{q} (1 - X_{ab}) d_{ab} P_{ab} \lambda_a + \alpha \sum_{a=1}^{p} \sum_{b=1}^{q} (X_{ab} S_b + (1 - X_{ab}) S_b P_{ab} \lambda_a) \]

\[ + \sum_{a=1}^{p} \sum_{b=1}^{q} (S_b + Bf(t)) S_b P_{ab} \lambda_a \] (5)
Subject to storage constraints
\[
\sum_{b=1}^{q} X_{ab}S_b \leq C_a \quad a=1,2,\ldots,p
\] (6)

To solve this optimization process we simplify Eq.(5) and obtain
\[
\min \left( \sum_{a=1}^{p} \sum_{b=1}^{q} (d_{ab}P_a\lambda_a + \alpha S_bP_a\lambda_a - X_{ab}(d_{ab}P_a\lambda_a + \alpha S_bP_a\lambda_a - \alpha S_b)) \right)
\]

**Algorithm 1.** EER approximation algorithm – runs on origin server.

for every proxy server \(a \in \{1, 2, \ldots, p\}\)  
Calculate \(f_{ab}\) and \(r_{ab}\) for each object \(b \in \{1, 2, \ldots, q\}\);  
Sort \(r_{ij}\) in descending order and record corresponding indices in array \(h\);  
Calculate total cost of system \(\text{TotalCost} = \text{TotalCost} + \sum_{b=1}^{q} k_{ab}\);  
Keep a record of the weighted total cost for a simple pull system  
\[\Delta_{pq} = \Delta_{pq} + \sum_{b=1}^{q} k_{ab}\]
for index \(i \in \{1, 2, \ldots, q\}\);  
\(b = h[\text{index}]\);  
if \((r_{ab} > 0 \text{ AND } s_b \leq \text{available storage on } i)\)  
Put \(b\) into \(\text{PushSet}_a\);  
\(\text{TotalCost} = \text{TotalCost} - f_{ab}\);  
Compress the content and store in proxy server \(a\);  
else if \((r_{ab} > 0 \text{ AND } s_b > \text{available storage on } a)\)
if \((\Delta_{pq} - f_{ab} < \text{TotalCost AND } s_b \leq c)\)  
Keep removing the last element in \(\text{PushSet}_a\) till there is enough room for \(b\);  
Update \(\text{TotalCost}\);  
Put \(b_j\) into \(\text{PushSet}_a\);  
end if  
else if \(r_{ab} \leq 0\)  
Break;  
end if  
end for

**Algorithm 2.** DEER approximation algorithm – runs on each proxy server.

Initialize the distance \(d_{ab}\) for each object \(b\) as the distance from origin server to local;  
Initialize the location of each object \(b\) as origin server;  
Calculate the \(f_{ab}\) and \(r_{ab}\) for each object \(b\);  
Sort \(r_{ab}\) in descending order and record corresponding indices in array \(h\);  
Calculate the \(\text{PushSet}_i\) according to \(r_{ab}\) and current \(d_{ab}\);  
Update the location for object \(s\) in \(\text{PushSet}_a\) as local;  
Inform cooperative-proxy server about current \(\text{PushSet}_a\);  
While \(\text{receive } \text{PushSet}_a\) ’ from cooperative proxy server \(a\)’  
For each objects \(k\) in received cooperate-proxy server’s \(\text{PushSet}_a\)’  
If distance from proxy server \(a\)’ to local is nearer than \(d_{ab}\);  
Update \(d_{ab}\) as the distance from proxy server \(a\)’ to local;  
record the location of object \(k\) as proxy server \(a\)’;

Calculate the storage available storage space on as proxy server \(a\)’;  
If storage space greater than received content size  
Receive the content from \(d_{ab}\) then compress the content and store in proxy server \(a\)’;  
End if  
End if  
End for  
For each objects \(b\)
If (Location is proxy server \(a\)’) AND (Not in \(\text{PushSet}_a\) ’)  
Update the distance \(d_{ab}\) as the distance from origin server to local;  
Update the location of object \(b\) as origin server;  
End if  
End for  
Recalculate the \(f_{ab}\) and \(r_{ab}\) for each object \(b\);  
Recalculate the \(\text{PushSet}_a\) according to updated \(r_{ab}\) and current \(d_{ab}\);  
Inform cooperate-proxy server updated \(\text{PushSet}_a\);  
End while

7 CONCLUSIONS

We have analysed the problem of replica selection and placement that determines which contents should be pushed to which proxy servers and which contents should be pulled on demand for an optimal content delivery over cloud storage. Our work is first considering the both bandwidth usage in the network and latency for the optimization of content delivery using cloud storage services. We have replica and formulated this push-pull content delivery problem and have designed an EERA (Enhanced Efficient Replication Algorithm) for our optimization problem. The proposed work focused on reducing the bandwidth usage, traffic, storage cost, delay access and latency also the performance cost and access time for storing and content delivery and yield the better results.

8 REFERENCES


[23] Lingfang Zeng, Bharadwaj Veeravalli, Qingsong Wei, ” Space4time: Optimization latency-sensitive content service in cloud”, Elsevier 2014.