ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

INTRA CLOUD LOAD BALANCING SCHEMES FOR EFFICIENT DELIVERY OF CDN SERVICES

P. Siva Rajesh*
[*] Dept of CSE, S.R.K.R engineering college, Bhimavaram, AP, India

DOI: 10.5281/zenodo.259590

ABSTRACT

We formulate the price optimization trouble with accurate cost models and QoS needs and reveal that the monthly cost is often as little as 2.62 $ for any small Site. Content supply set-up (CSS) using storage clouds have lately began to emerge. However, existing focus on replica placement in CSS doesn't readily apply within the cloud. Within this paper, we investigated the joint problem of creating distribution pathways and placing Server replicas in cloud CSS to reduce the price incurred around the CSS providers while satisfying QoS needs for user demands. When compared with traditional CSS, storage cloud-based CSS have the benefit of cheaply offering hosting companies to Content providers without owning infrastructure. We create a suite of offline, online-static internet-based-dynamic heuristic algorithms that take as input network topology and work information for example user location and ask for rates. Then we assess the heuristics via Web trace-based simulation, and reveal that our heuristics behave not far from optimal under various network conditions.

KEYWORDS: Offline and Online algorithms, Content Supply Set-up (CSS), Quality of Service(QoS).

INTRODUCTION

Two possibilities arise in the emergence of storage cloud services. First, it's possible to develop a CSS serving others without the cost of owning or operating geographically spread data centers. Storage cloud providers charge their clients by their storage and bandwidth usage following a utility computing model. Second, small Internet sites can take shape their very own global CSS simply by becoming customers of multiple storage cloud providers operating in various continents and locales. Storage price is measured per GB per unit some time and bandwidth price is measured per GB transferred. Bandwidth cost further includes two components: upload cost for incoming data and download cost for outgoing data [1]. A cloud CSS also multiplexes sources among multiple customers/Internet sites just like a traditional CSS. Quite simply, cloud CSS can offer similar functionality as traditional CSS, but without really owning any infrastructure. To efficiently manage a cloud CSS, intelligent replica placement and user redirection strategies are needed. This topology might be not the same as the actual network topology. Thus, replica placement in cloud CSS is really a joint problem of creating distribution pathways and replication. However, storage clouds charge different prices for uploading and installing, which necessitates the edge to become directed. What this means is that just selecting some replica sites is not adequate enough we have to specify the replication directions. Request redirection is an essential part of replica placement. Any replica placement formula necessitates the specs from the corresponding request redirection strategy. However, merely a subset of those pathways satisfies QoS needs for user demands that are attracted as dashed lines [2]. The all inclusive costs with this solution include storage, upload and download cost at C0, C1, C3 and C4. For any replica site that serves user demands, its upload price is suffered by incoming visitors to provision Content in the node, and it is download price is suffered by outgoing visitors to serve user demands. Within this paper, we advise a suite of algorithms for replica placement for any cloud CSS. We first produce an Integer Programming (IP) formulation of the problem and prove that it's NP-hard. Then we propose two teams of heuristics to put replicas: offline and static algorithms according to past user request patterns an internet-based-static an internet-based-dynamic algorithms triggered by each request.

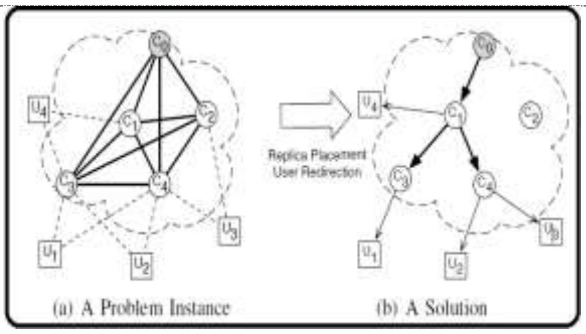


Fig.1.Overview of the scenario

RELATED WORK

Content placement should be efficient to achieve better availability of content to users located in different geographical locations. Some of the previous works are as follows
Li et al. proposed a placement algorithm in [3] based on the assumption that the underlying topologies are trees, and modeled it as a dynamic programming problem. The algorithm was actually designed for Web proxy cache placement, and it is also applicable for Web replica placement.

Qiu et al. [4] having evaluated a number of algorithms and comes up with a greedy algorithm which offers best performance. The idea behind the greedy algorithm is as follows. Suppose we need to choose P replicas among Q potential sites. We choose one replica at a time. The greedy algorithm can be implemented in iterative fashion. At each stage it selects the Q potential sites which are optimal. We compute the cost associated with each site according to the assumption that the site is accessed by all clients converge at that site, and pick the site that yields the lowest cost. In the second iteration, we search for a second replica site which, in conjunction with the site already picked, yields the lowest cost. In general, in computing the cost, we assume that clients direct their accesses to the nearest replica (i.e., one that can be reached with the lowest cost). We iterate until we have chosen P replicas.

Charikar et al. proposed a 4-approximation algorithm for solving the minimum K-median problem in [5]. This is so far the best known approximation algorithm in the worst case bound for the metric K-median problem in which the distance function c is non-negative, symmetric and satisfies the triangle inequality.

Kalpakis et al. [6] proposed an algorithm which considers all three costs (retrieval, update and storage). In this paper they considered only tree topology. But none of the works is related to provisioning cost between replica sites is relevant to the replication direction.

Tang et al.[7] investigates the QoS-aware replica placement problems for responsiveness QoS requirements. In this paper they considered two classes of service models: replica-aware services and replica-blind services. In replica-aware services, the servers are aware of the locations of replicas and can therefore optimize request routing to improve responsiveness. We show that the QoS-aware placement problem for replica-aware services is NP-complete. In blind services, the servers are not aware of the locations of replicas or even their existence. As a result, each replica only serves the requests flowing through it under some given routing strategy. Efficient algorithms are proposed to compute the optimal locations of replicas under different cost models.

The solutions provided upto now are static in nature based on the request history the placement decisions are taken. Bartolini et al. [8] proposed a centralized heuristic which is a unique approach for modeling the replica placement of content in cloud storage. Vicari et al. [9] optimized replica placement and traffic redirection in order to balance request load on replicas.

METHODOLOGY

In the web based setting, we assume all input parameters to replica placement are known except future demands. What this means is that people are only able to consider demands instantly and our decision ought to be made according to each request. A formula could be either static or dynamic. The offline algorithms are static anyway, as the online algorithms are generally static, using past user request patterns to derive replica placement or dynamic, making placement decisions reacting to real-time user demands [3]. We formulate the offline problem of replicating an internet site replica/segment located by an origin server C0 at multiple storage sites for everyone all finish user demands. To be able to satisfy QoS needs for finishing user demands, we have to limit this distance inside a needed QoS distance Q. We don't put bandwidth capacity like a hard constraint on replica sites. The goal of the replica placement problem is to locate a replication strategy minimizing all inclusive costs by provisioning a subset from the cloud sites so that each finish user is assigned to a single site and also the QoS needs for finishing user demands are satisfied. We produce an NP-hard evidence of this issue inside a technical report and reveal that the issue can't be approximated with factor much better than O log (m), where m is the amount of users. The Replica Provision operation for any cloud site Cj includes installing a duplicate from the proper site Ci and uploading and storing the replica on Cj [9]. The 2nd operation is User Redirection or User Assignment. Within the offline setting, all demands of the user are forwarded to the very best replica site. In the web based setting, each user request is redirected individually. The Greedy Site (GS) formula is adopted from your approximation formula for that Set Covering Problem. The possibility users of the site would be the users who're inside the QoS distance of the site although not yet assigned. We open this website, and then look for the following best site to spread out until all users are allotted to a website. Within the formula, we assign users to sites using the cheapest cost and open new sites if required. In the web based setting, it's possible to use a result of past request patterns to create replica placement decisions using offline algorithms, and just redirect user demands towards the already opened up sites. We consider users using the least quantity of potential sites first. We present three online heuristics. We first present the fundamental formula Greedy Request where each request is processed upon arrival. We've the next three online schemes: Greedy Request Only, Greedy Request with Reallocation and Greedy Site Only. One method to handle online demands is by using Greedy Request (GR) only, and open sites if needed inside a dynamic fashion. Observe that site opening operation is definitely triggered through the first request of the user. We use Greedy Site (GS) according to recent user demands to derive some sites, and provision replicas in it. Upon the appearance of a brand new request, we execute Greedy Request (GR) dynamically. However, we are able to fix the group of open sites based on Greedy Site (GS) according to request history, and redirect all future demands to existing open sites only. The Replica Provision process thus remains static [7]. User Redirection within this situation is slightly different: when you will find reallocated open sites inside the QoS distance, we assign the consumer towards the site using the least cost otherwise; we assign the consumer towards the nearest site no matter cost. The data on potential replica sites is obtained from the iPlane Internet PoP topology. To put a node, we start looking up its Ip within the GeoLite city database and convert the Ip to the corresponding pair. Using finish user IPs from server access traces, we map finish users to the same metric field using the origin server C0 and also the cloud sites. Actually, the option of distance metric doesn't change up the performance in our algorithms any distance metric that is capable of doing describing the QoS requirement is relevant. Observe that Greedy Request (GR) can be used either an offline or perhaps an online formula, therefore we include it for comparison [10]. For that two heuristics requiring site reallocation, namely, Greedy Request with Reallocation (GRP) and Greedy Site Only (GSO), we first allocate sites according to conjecture of user request patterns. We make use of the immediate past record being a symbol of future user behavior. The 3 online schemes in decreasing performance or growing cost order are GSO, GRP and GRO.

PROPOSED SYSTEM

Earlier frameworks took care of the issue of reproduction situation effectively utilizing disconnected and online calculations. In doing as such they considered Site, User and Request measurements. At times a solitary client may produce heaps of action including access, asks for, transfers and downloads. A heuristic forecast based load adjusting that considers both capacity and data transmission expenses of CDN servers is essential for an element provisioning of CDN servers. Some of the time it is important for a heuristic choice module that can progressively turn off less stacked server and move customers to residual servers and switch it back on if there should be an occurrence of increment in general load. So we propose the accompanying expectation based load adjusting module based that can trigger server hubs in view of their heaps This approach conveys best execution in overseeing cloud servers and decreasing general expenses.

Later on, we plan to grow our detailing to expressly consider both data transfer capacity and capacity limit at a cloud site.

In actuality, inertness can fluctuate in light of clog, system disappointments, course changes and so forth, which is not unequivocally tended to in this paper. We plan to research system to manage idleness changeability and in addition other part of the. For certain web content, for instance video, measurements, for example, throughput is more imperative.

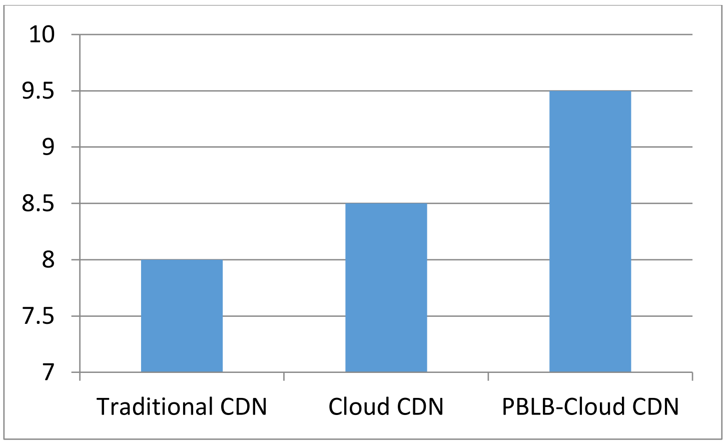
Algorithm 1 Prediction-based Load Balancing

```
1: nodes ← Servers with utilization statistics
2: while n ∈ nodes do
3:     if n is triggered then
4:         triggers ← triggers U {n}
5:     end if
6: end while
7: Sort the imbalance scores of triggers in descending order
8: while n ∈ triggers do
9:     if attractiveness(n) > threshold then
10:        nodeToMigrate←node
11:    end if
12: end while
13: dest ← smallestLoad(nodeToMigrate)
```

Algorithm1.Predition-based load Balancing

RESULT ANALYSIS

In this paper web-based simulation is used to analyze the performance of cloud CDNs. Here three types of cloud CDNs are used they are: traditional CDNs, cloud CDNs and prediction-based load balancing (PBLB)-cloud CDNs. Here in this paper bandwidth cost and storage cost are considered, comparing to traditional and cloud CDNs our proposed PBLB-cloud CDNs are outperformed. And it helps in reducing cost by progressively turn off less stacked server and move customers to residual servers and switch it back on if there should be an occurrence of increment in general load. By this heuristic model the intelligent replica placement and user redirection strategies along with Owner control of CDN's for cost control, leading to better and affordable distribution services. A sample bar graph is drawn to show the performance of the three CDNs.



Performance graph

CONCLUSION

We formulated the issue being an Integer Program and presented various offline an internet-based greedy heuristics. We intend to investigate mechanism to cope with latency variability along with other part of the SLA. Within this paper, we investigated the issue of placing Server replicas kept in storage cloud-based CSS together with building distribution pathways included in this to reduce the price incurred around the CSS providers while satisfying QoS needs for user demands. We've the next three online schemes: Greedy Request Only, Greedy Request with Reallocation and Greedy Site Only. For several content, for instance video, SLA metrics for example throughput is much more important. We evaluated their performance from the optimal via Web trace-based simulations. Later on, we intend to expand our formulation to clearly consider both bandwidth and storage capacity in a cloud site. The truth is, latency can differ due to congestion, network failures, route changes etc, which isn't clearly addressed within this paper.

REFERENCES

[1] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, “iPlane: An information plane for distributed services,” in Proceedings of the 7th symposium on Operating systems design and implementation. USENIX Association, 2006, pp. 367–380.

[2] P. Radoslavov, R. Govindan, and D. Estrin, “Topology-informed Internet replica placement,” Computer Communications, vol. 25, no. 4, pp. 384–392, 2002.

[3] X. Jia, D. Li, X. Hu, W. Wu, and D. Du, “Placement of web-server proxies with consideration of read and update operations on the internet,” The Computer Journal, vol. 46, no. 4, p. 378, 2003.

[4] L. Qiu, V. Padmanabhan, and G. Voelker, “On the placement of web server replicas,” in IEEE INFOCOM 2001 Proceedings, vol. 3, 2001.

[5] M. Charikar, and S. Guha. Improved Combinatorial Algorithms for the Facility Location and K-Median Problems. In Proc. of the 40th Annual IEEE Conference on Foundations of Computer Science, 1999.

[6] K. Kalpakis, K. Dasgupta, and O. Wolfson, “Optimal placement of replicas in trees with read, write, and storage costs,” IEEE Transactions on Parallel and Distributed Systems, pp. 628–637, 2001.

[7] X. Tang and J. Xu, “QoS-aware replica placement for content distribution,” IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 10, pp. 921–932, 2005.

[8] N. Bartolini, F. Lo, and P. Petrioli, “Optimal dynamic replica placementcin content delivery networks,” in Proc. IEEE Int. Conf. on Networking,2003, pp. 125–130.

[9] F. Lo Presti, C. Petrioli, and C. Vicari, “Dynamic replica placement in content delivery networks,” in 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005, pp. 351–360.

[10] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, “Search and replication in unstructured peer-to-peer networks,” in Proceedings of the 16th International conference on Supercomputing, 2002.