Local Area Redirection Policies for a Lightweight Distributed Desktop Webgrid

Marie Yvette B. de Robles[1] and Jaderick P. Pabico[2]
[1]*mybderobles@uplb.edu.ph*, [2]*jppabico@uplb.edu.ph*
[1]Institute of Computer Science, University of the Philippines Los Baños
PHILIPPINES

Abstract *– With the latest innovations and trend towards personalizing users' web browsing experience, the web has been increasingly dominated by dynamic contents. Solutions abound for improving the performance of static contents today. However, delivering dynamic content remains a challenge due to the many dependencies involved in compiling the content, specifically personalized ones. This paper presents two local area redirection (LARD) policies, the round-robin (RR) and the least busy server first (LBSF) for a cheap, off-the-shelf, local lightweight distributed grid composed of desktop PCs (webgrid). Each LARD algorithm redirects web requests away from a busy server in the webgrid, developed as an alternative architecture for both static and dynamic-content Web sites. The architecture was designed such that performance as characterized by client access time and resource utilizations is optimized during overload conditions, such as flash crowd and time-of-day effects. Through an online simulation technique, we explored the performance savings achieved by the LARD algorithms. Results showed that the end-to-end client delays were improved for both simple and complex jobs. For simple jobs, fewer computers are enough to service the requests at the normal response time. Moreover, the performance continuously improves as the number of computers increases for complex jobs. Results also showed that RR is a better LARD policy than LBSF. LBSF has an added overhead required for the online inventory of the least busy servers in the webgrid.*

Keywords *–*Web servers, request redirection, local area networks, webgrid, desktop PCs, performance evaluation

I. INTRODUCTION

Content Delivery Networks (CDNs), coupled with mirroring and caching strategies, aims to reduce the network latency between the clients and the data they are accessing. The usual solution implemented by big entities with worldwide scope such as Google and Yahoo is by using a distributed server architecture that is transparent from the user while preserving one virtual URL interface. With this architecture, clients are redirected to the geographically closest server wherein popular URLs are optimally placed on replicas closer to some hot spots. This solution provides both scalability and transparency.

Distributed server architectures are specially designed to improve the Quality of Service (QoS) of static content such as image files where network latency,and hence transmission time are critical components. However, with the latest trend towards personalizing users' browsing experience, web content has become increasingly dominated by dynamic content. Because dynamic content is computationally intensive, it has different demands on the required server resources as compared to static content. Consequently, the server selection mechanism designed for static content may not be optimal for dynamic content. Dynamic content takes into consideration both network latencies and server loads. In some cases, it may make sense to forward a dynamic-content request to the spatially closest server.Inother cases, it may be better to forward it to the server that is geographically furthest if it has the lowest sum total of network latency and expected server processing time.

Web architectures today are manually configured and cannot automatically adjust to changing workloads such as those caused by time-of-day (TOD) effects and flash crowd arrivals (FCA). TOD is the diurnal variation in traffic observed at most Web sites which usually vary between a factor of 2 and 20 throughout the day, whileFCA is the unexpected flood of users accessing a Website at an unexpected time or unexpectedly high magnitude. Allocating resources enough for an average workload will make the performance suffer significantly when loads exceed the capacity. On the other hand, allocating resources which are able to handle the peak workload will result to poor resource utilization for most of the time. Hence, allocating the optimal number of needed servers would be advantageous. For instance, servers which are seldom used can be powered off leading to significant power cost savings. However, the challenge in designing an optimal resource allocation policy is to minimize server resources without affecting the user-perceived QoS.

Several researches have been conducted to improve the performance of both static and dynamic content

P-ISSN 2362 – 8022 | E-ISSN 2362 – 8030 | www.apjeas.apjmr.com

server policies. In static server policies, the concern is on reducing the end-to-end network latency by redirecting a request to the closest server. These architectures implement two models: (1) a shared server utility model where many services share a server at a time, or (2) a full server utility model where each server offers one service at a time. In server selection for dynamic content, caching has been proposed as one of the best solutions which can occur at either the client side or at the server side[1].

There are also different redirection algorithms proposed to address both static and dynamic content server policies. However, most of them focused on client-side mechanisms which considered network as the primary bottleneck. Nevertheless, recent developments showed that due to the increase of dynamic-content Web sites and the existence of high interconnection links, network latency is not the main issue anymore but the internal processing brought about by dynamic-content Web sites. Today, there are few redirection algorithms which tried to solve this problem. One of these is the wide-area redirection (WARD) algorithm where requests are redirected to the best server, which could either be geographically located locally or globally (see Figure 1).

The purpose of WARD is to minimize the total networking and server processing delays. However, WARD may only apply to large data centers that cater millions of requests per day such as Google and whose hosting architecture comprises of clusters that are mostly located remotely. A typical web host may not need too many servers scattered globally and may rely on some other resources which may be already available locally. Furthermore, WARD failed to consider the same demand on static-content Websites.

This paper proposes a prototype distributed desktop webgrid as a low-cost, off-the-shelf architecture for static and dynamic-content Web sites. The aim of the webgrid is to optimize the perceived performance of the server characterized by client access time and resource utilizations even during overload conditions. This architecture considers personal computers, particularly those which are only used in simple tasks, as key elements in improving the performance of dynamic-content Web sites. A Local Area Redirection (LARD) per-request (or per-query) policies were used and evaluated. These policies, namely round-robin (RR) and least busy server first (LBSF), redirect requests (or queries) away from an overloaded server to personal computers (PC) located locally (see Figure 2).

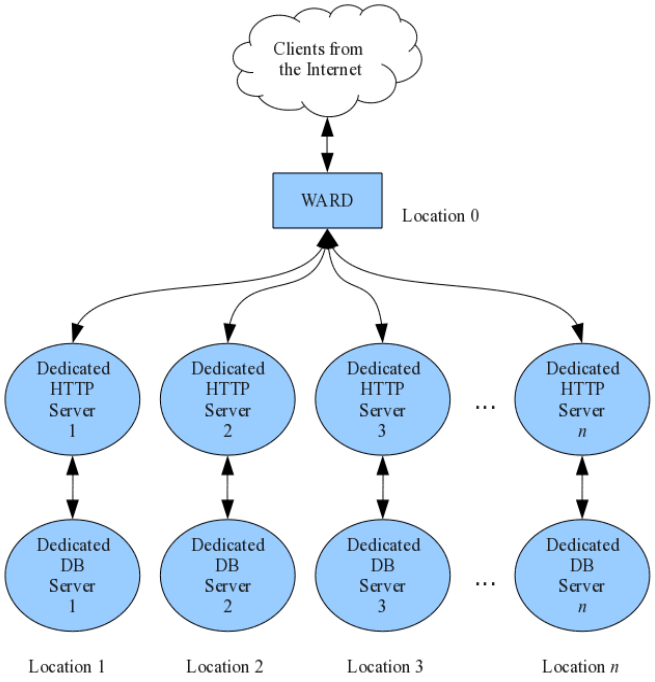

Fig. 1. The conceptual framework of a WARD architecture. Dedicated HTTP and DB servers maybe geographically located differently.

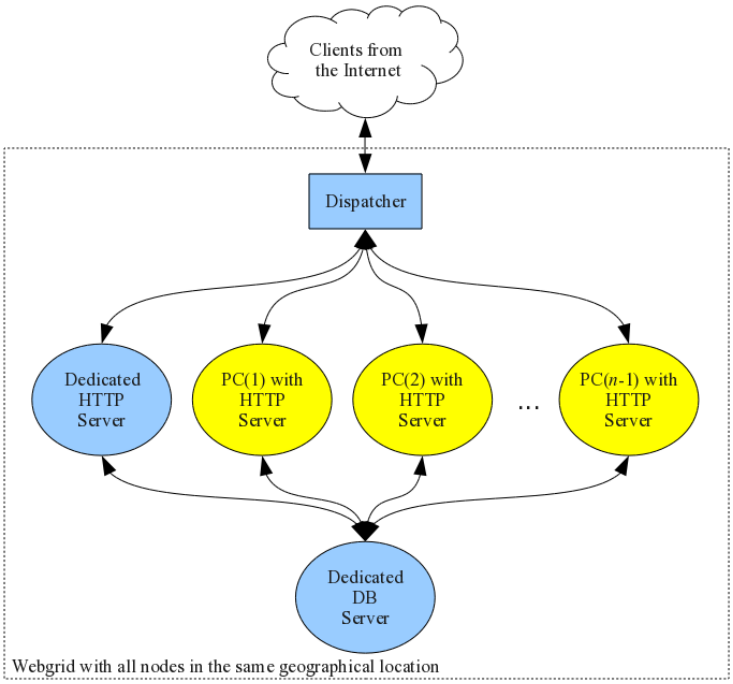

Fig. 2. The conceptual framework of the proposed webgrid architecture composed of the dispatcher, dedicated HTTP and DB servers, and a grid of HTTP servers composed of desktop PCs. All machines are geographically located at the same place.

P-ISSN 2362 – 8022 | E-ISSN 2362 – 8030 | www.apjeas.apjmr.com

II. REVIEW OF LITERATURE

Several researches have been conducted to improve theperformance of static server allocation policies. Some of the most widely used strategies include Content Delivery Networks (CDNs) [2, 3, 4], mirroring [5, 6], and caching [7, 8, 9]. In CDNs, content is replicated on CDN servers which are located close to the final users, and user requests are redirected to the closest replica. Caching involves the passive replication and storage of content passing through a node. Mirroring involves the use of storage nodes, known as mirror sites deployed across different geographical locations, that is a complete replicate of the origin server. On the other hand, caching of dynamic fragments such as those discussed in [10, 11, 12] addresses server selection for dynamic content. Caching can occur at either the client side or at the server side. At the client side, there are expiration times set using cookies while at the server side, there are cached pages that expires on receiving database update queries. One common disadvantage of dynamic-content caching is it is not backward compatible. It requires the web service developers to adhere to a new set of strict guidelines which needs an enormous adjustment to their application.

In the area of redirection mechanisms, several researches have focused on client-side mechanisms such as requestredirection in CDNs [13, 4, 14], server selection techniques [15, 16], caching [17], mirroring, and mirror placement [18, 19]. These techniques tried to solve a perceived problem that the network is the primary bottleneck. However, with the increasing trend towards dynamic-content Web sites, where server processing times are more important than network latency, these mechanisms may not be applicable.

Thus, recent developments have been made to focus on the internal complexity brought about by dynamic-content Web sites such as the one discussed by [1]. This mechanism makes use of Wide-area Redirection(WARD) where requests are redirected to the best server, which could either be located locally or globally, such that total networking and processing delays are minimized.

Several architectures have been designed to improve the performance of web content, whether static or dynamic. However, these architectures comprise mainly of servers as elements, which could either be hosting one content or different contents. These architectures are also designed for large data centers only where servers are located globally and requests are redirected via WARD algorithms. Furthermore, they also failed to consider the increasing complexity of static-content Web sites.

On the other hand, there has been no proposed architecture implementing a LARD algorithm, which makes use of personal computers that are connected locally to the server as elements in the webgrid. Thus, this paper proposes a prototype distributed desktop webgrid which aims to optimize the performance of Web sites, as characterized by client access time and resource utilizations, even during overload conditions. This is done via a Local Area Redirection (LARD) per-request (or per-query) redirection algorithm that minimizes the total networking plus server processing delay.

III. METHODOLOGY

The webgrid architecture was implemented using Apache as web tier, Java Servlets as application tier, and MySQL as database tier.

Our Web architecture was presented in which each node is hosting a complete replica of the application. Also, the node given the request should act both as the application and the database server. Thus, a dispatcher was placed only at the application tier (see Figure 2). The dispatcher was implemented with two different LARD policies. To illustrate this architecture, let us consider the requests of a dynamic web session. First, a client request arrives at a dispatcher which in turn dispatches the request to a server. The dispatcher will have to decide for the best node which will serve as the application server that will resolve all the dynamic fragments embedded within the client request by generating relevant database queries. Finally, the application server collects all the responses to the database queries and returns the page directly to the client. The architecture was evaluated and implemented as follows:

A. Servlets

Servlets are computer programs that provide services to clients. Two different servlets were implemented, simple and complex. A simple servlet performs simple tasks for the clients and does not require long and complex database queries. A complex servlet performs complex tasks for the clients and usually requires more computational tasks and complex database joins. For this experiment, a complex task requiresabout 500 more server resources than a simple task.

B. LARD Algorithm

Two algorithms for local-area redirection were used, the RR and the LBSF. In RR, requests are assigned by

P-ISSN 2362 – 8022 | E-ISSN 2362 – 8030 | www.apjeas.apjmr.com

the dispatcher to local servers in a circular order, while in LBSF, requests are assigned to the server which currently has the least number of loads. LBSF is more complex than RR since the dispatcher has to be updated online regarding the current load of the servers. However, the RR algorithm does not consider the current load of a server which is an important factor that contributes to the performance of a machine.

C. Performance Evaluation

We showed that these architectures were able tooptimize the performance of the server as characterized by the average total access delays perceived by clients. This was done through an online technique where a program was created to generate requests of increasing workloads. These requests are fed to the architecture without redirection, and to the different architectures configured with a redirection policy. We calculated the total response time of the different LARD implementations and compared them to the total response time of that which does not implement redirection. The Total Response Time (TRT) can be computed as

$$TRT = RT_s + RT_d + T \quad (1)$$

where RT_s is the average response time of servers, $RT_d$is the average response of the dispatcher, and T is the delay caused by the traffic between communication lines.

We evaluated the performance of the webgrid by subjecting the dispatcher to varying workloads, characterized by the number of requests it received per second. Here, we looked at workloads with values 2, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, and 200 requests per second (rps). For each workload, we observed the effect of increasing the number of PCs n involved in the grid on TRT. We used n = 2, 4, 6, 8, 10, 12, 14, and 16 PCs. We have also compared the effects of the LARD policy on TRT at certain combination of workload and n.

IV. RESULTS AND DISCUSSION

Using our experimental testbed, we first validated theefficacy of LARD in improving the performance ofservers implementing either the RR or the LBSF policies and compared the performance against a single server (S1) architecture (see Figure 3).

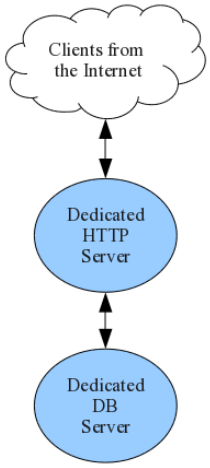

Fig. 3. The framework of a single server HTTP service.

Figures 4a-4b show the performance achieved byS1 and the webgrid with RR and LBSF as LARD policies given simple jobs. The performance of the webgrid implementing either RR or LBSF is better than the S1. This highlights the fact that S1's workload is heavy; hence, it is of benefit to redirect a part of its load to the webgrid which can perform the same job. Resultsshow that as the workload is increased, the utility of the webgrid becomes beneficial. We computedthe percent decrease in TRT (see Equation 2) caused by the webgrid. In the equation, RT_1 is the response time of S1, and $RT_n$is the response time of the webgridwith $n - 1$ participating PCs.

$$PD = \frac{RT_1 - RT_n}{RT_1} \times 100 \quad (2)$$

The webgrid with $n = 2$ and RR as the dispatcher algorithm, PDs of 10.93% to 96.01% were observed for all workloads. However, when the LBSF policy was used, the PD was decreased by 106.90% for the lowest workload of 2 rps, but increased to as high as 62.28% for the highest workload of 200 rps.

The same data were gathered for the architecture implementing RR and LBSF with n = 4, 6, 8, 10, 12, and14 (see Figures 4a-4b). The webgrid implementing RR and LBSF with four computers yielded an even better result compared to that with only two computers. For RR, a 96.90% decrease in response time was experienced for the 150 rps workload, and a 97.85% decrease for the 200 rps workload. For LBSF, an 88.62% and a 90.83% decrease in response time was experienced for the 150 rps and 200 rps workload respectively.

P-ISSN 2362 – 8022 | E-ISSN 2362 – 8030 | www.apjeas.apjmr.com

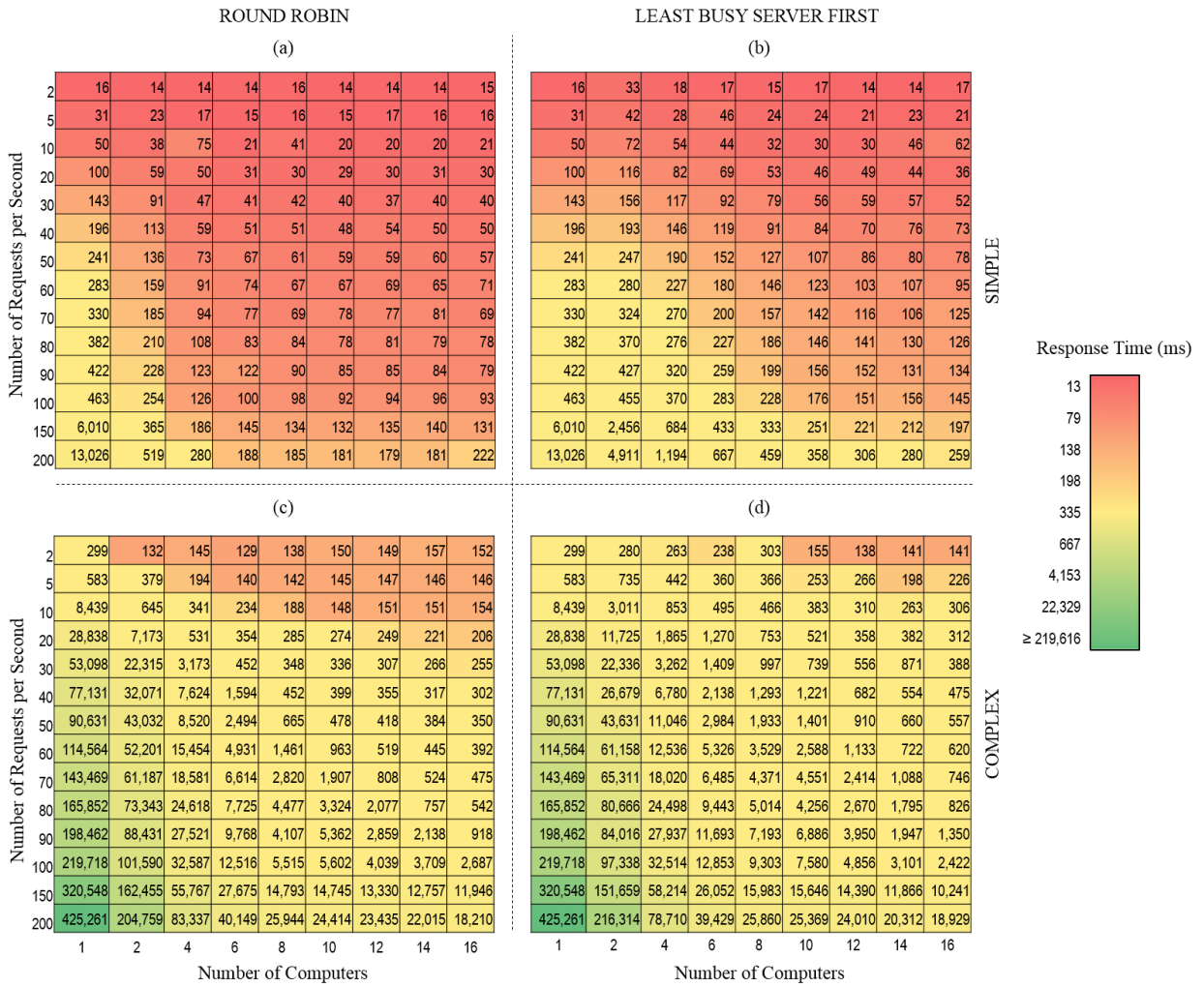

Fig. 4. Mean Response Time of the Webgrid Implementing (a) Round Robin Given Simple Jobs,(b) Least Busy Server First Given Simple Jobs, (c) Round Robin Given Complex Jobs, and (d) Least Busy Server First Given Complex Jobs onIncreasing Number of Requests per Second and Increasing Number of Servers Participating in theWebgrid (n=1000)

The average response time, however, did not improve with the further increase in n. The performance achieved by the architecture with n = 4 is comparable to the performance of the architecture with n> 4. This is because the job is not complex enough that n = 4 is already an optimal webgrid size. The moment a new set of requests arrive, the four PCs have already finished the current jobs sothe computers do not really get busy.

Because of this, we next observed the performance of the webgrid tasked with complex jobs. Again, we compared the performance of the webgrid against the S1. Like the previous experiment, the performance of LARD implementing RR and LBSF is better than the S1 with a much pronounced difference (see Figures 4c-4d). We first observed the architecture with n = 2. Under the RR policy, there is a 55.72% decrease in the response time given a workload of 2 rps and a 35% decrease for a workload of 10 rps. On the other hand, a lower percent of decrease in the response time can be observed for the higher workloads compared to the previous setup. For the 100 rps workload, there is a 92.78% decrease in response time while for the 200 rps workload, a 92.47% decrease was obtained.

Experiments implementing the LBSF policy provide a similar pattern as that of the RR policy. There is a 99.82% and 97.55% decrease in the response time for the workload of 5 rps and 10 rps respectively. For the higher workloads of 100 rps and 200 rps, a 92.11% and 91.70% respective decrease in the response time were observed.

P-ISSN 2362 – 8022 | E-ISSN 2362 – 8030 | www.apjeas.apjmr.com

The performance of the webgrid continuously improves as$n$ is increased. For a workload of 5 rps, the performance was maintained to a near 100% response time decrease across all n. For 10 rps, the performance was also improved to a near 100% decrease. The results were much evident on higher loads. For a grid with n = 10, a 98.96% decrease (RR) and a 99.17% decrease (LBSF) in response time wereobserved with a workload of 100 rps. For the 200 rps workload, the response time was reduced to 98.52% and 99.03% for RR and LBSF respectively.

Lastly, we comparedthe performance of RR and LBSF. Figure 4 shows that RR is a better algorithm than LBSF. However, due to the limitations of the prototype architecture, it is expected that the results may vary. First, in our model, a global packet collision is experienced. That is, communication between a server and the dispatcher will interrupt the passage of the requests to other computers. And due to the added task in LBSF in updating the dispatcher every time it finishes a job, thus increasing global packet collision, it is expected that packet collisions are more frequent in LBSF. Second, computers used in this experiment are homogeneous. In effect, computers are expected to finish the requests at the same time and thus, choosing the least busy server would not matter much. Furthermore, the mean response time of the dispatcher for both algorithms is zero on the average. Thus, the added complexity brought by the dispatcher is insignificant and would not affect the performance of the architecture.

IV. SUMMARY AND CONCLUSION

A prototype distributed desktop webgrid was developedto improve the performance of both static and dynamic-content Web sites, characterized by client access time and resource utilization, even during overload conditions. Through an off-line simulation technique, performance savings achieved by the proposed algorithm was explored. Two different algorithms were implemented and evaluated, the RR and the LBSF.

With the new architecture, utilization of resources, especially servers or computers were optimized. Allocation of additional resources to handle the peak workload caused by Time-of-Day effects and Flash crowd arrivals are no longer necessary. Instead, computers such as those belonging to secretaries, which are used only for simple tasks such as typing, replaced the task of the servers in providing services to clients. Thus, a significant power and other resources savings were achieved.

Results also showed that LARD can be used to avoid short-term bottlenecks due to flash crowds arrivals and time-of-day effect. The end-to-end client delays were improved for both simple and complex jobs. For simple jobs, few computers are enough to service the requests at the normal response time despite the increasing workload. The effect becomes more obvious as you increase the workload. For complex jobs, the end-to-end client delays continuously decrease as the number of computers increases. But unlike the former, the effect is more obvious on lower workloads. Availability of more machines, however, achieved the same result in both low and high workloads.

The two algorithms were also compared. Results showed that RR is a better algorithm than LBSF. However, due to the limitation of the model, the results were inaccurate. But it is important to note that the experiment represents the worst case performance of the desktop distributed webgrid. Thus, a better result is expected when the experiment was performed in the real-world setting. Also, our model is more unfavourable in LBSF since traffic is more existent in LBSF. This is because of the added task of the servlets to update the dispatcher whenever they finish a job. Thus, in the actual implementation of our webgrid, the boundaryfor LBSF is expected to increase.

IV. RECOMMENDATIONS

The experiments were done on homogeneous computers.The difference between the dispatcher algorithms RR and LBSF might be more apparent when heterogeneous computers are involved. Other algorithms or combination of algorithms can also be used to replace the LARD algorithms presented.

The experiments can be performed in the real-world settings to validate the methodology presented in this paper. The use of the actual webgrid will help obtain reasonable load and network proximity estimates, which could be refined over time to improve their accuracy, in response to changes in network.

REFERENCES

[1] S. Ranjan and E. Knightly. 2008. *High-performance resource allocation and request redirection algorithms for web cluster.*In Proceedings of the 2008 IEEE Transactions on Parallel and Distributed Systems (TPDS08), 19(9):1186-1200.

[2] D. Leong, T. Ho and R. Cathey. 2009.*Optimal content delivery with network coding.* In Proceedings of the 2009 IEEE International Conference on Information Sciences and Systems (CISS09), pp. 414-419(ISBN 978-1-4244-2733-8).

[3] Z. Lu, W. Fu, S. Zhang and Y. Zhong. 2008. *TRRR: A tree-round-robin-replica content replication algorithm for improving fastreplica in content delivery networks.*In Proceedings of the 2008 IEEE International Conference onWireless Communications, Networking and Mobile Computing (WiCOM08), pp. 1-4 (ISBN 978-1-4244-2107-7).

[4] F.L. Presti, C. Petrioli and C. Vicari. 2007. *Distributed dynamic replica placement and request redirection in content delivery networks*. In Proceedings of the 2007 IEEE International Conference on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS07), pp. 366-373 (ISBN 978-1-4244-1853-4).

[5] A. Gavrilovska, K. Schwan and V. Oleson. 2001. *Adaptive mirroring in cluster servers*.In Proceedings of the 2001 IEEE International Conference on High Performance Distributed Computing (HPDC01), pp. 3-13 (ISBN 0-7695-1296-8).

[6] A. Myers, P. Dinda and H. Zhang. 1999. *Performance characteristics of mirror servers on the internet.*In Proceedings of the 1999 Joint Conference of the IEEE Computer and Communications Societies (INFOCOM99), 1:304-312 (ISBN 0-7803-5417-6).

[7] A. Bhattacharjee and B.K. Debnath. 2005. *A new web cache replacement algorithm.*In Proceedings of the 2005 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM05), pp. 420-423 (ISBN 0-7803-9195-0).

[8] V.J. Sosa, G. Gonzales and L. Navarro. 2003. *Building a flexible web caching system.*In Proceedings of the 2003 Mexican International Conference on Computer Science (ENC03), pp. 60-65 (ISBN 0-7695-1915-6).

[9] Q. Zou, P. Martin and H. Hassanein. 2003. *Transparent distributed web caching with minimum expected response time.*In Proceedings of the 2003 IEEE International Conference on Performance, Computing, and Communications(PCCC03), pp. 379-386 (ISBN 0-7803-7893-8).

[10] S. Bouchenak, S. Mittal and W. Zwaenepoel. 2003. Using code transformation for consistent and transparent caching of dynamic web content. EPFL 200383.

[11] V. Holmedahl, B. Smith and T. Yang. 1998. *Cooperative caching of dynamic content on a distributed web server.*InProceedings of the 1998 International Symposium on High Performance Distributed Computing (HPDC98), pp. 243-250 (ISBN 0-8186-8579-4).

[12] B. Smith, A. Acharya, T. Yang and H. Zhu. 1999. *Exploiting result equivalence in caching dynamic content on a distributed web server*.In Proceedings of the 1999 Conference on USENIX Symposium on Internet Technologies and Systems (USITS99), 2:19.

[13] J. Kangasharju, K.W. Ross and J.W. Roberts. 2001. *Performance evaluation of redirection schemes in content distribution-networks.* Computer Communications, 24(2):207-214.

[14] L. Wang, V. Pai, and L. Peterson. 2002. The effectiveness of request redirection on cdn robustness.In Proceedings of the 2002 Symposium on Operating Systems Design and Implementation (OSDI02), 36(SI):345-360 (ISSN: 0163-5980).

[15] N. Kamiyama. 2006. *Optimum server selection in content distribution networks.*In Proceedings of the2006 Global Telecommunications (GLOBECOM06), pp. 1-6 (ISBN 1-4244-0356-1).

[16] T. Wu and D. Starobinski. 2008. *A comparative analysis of server selection in content replication networks.*In Proceedings of the 2008 IEEE/ACMTransactions on Networking (TNET08), 16(6):1461-1474 (ISSN 1063-6692).

[17] D. Karger, A. Sherman, A. Berkhemier,B. Bogstad, R. Dhanidina, K. Iwamoto, B. Kim, L. Matkins and Y. Yerushalmi. 1999. *Web caching with consistent hashing.*Computer Networks: The International Journal of Computer and Telecommunications Networking, 31(11-16):1203-1213 (DOI 10.1016/S1389-1286(99)00055-9).

[18] Y. Chen, R.H. Katz and J.D. Kubiatowicz. 2002. *Dynamic replica placement for scalable content delivery.*In Proceedings of the 2002 International

Workshop on Peer-to-Peer Systems (IPTPS02), 2429:306-318 (978-3-540-44179-3).

[19] S. Jamin, C. Jin, A.R. Kurc, D. Raz andY. Shavitt. 2001. *Constrained mirror placement on theinternet.*In Proceedings of the 2001 Joint Conference of the IEEE Computer and Communications Societies (INFOCOM01), 1:31-40 (ISBN 0-7803-7016-3).