Advances in Science and Technology
Research Journal
Volume 7, No. 20, Dec. 2013, pp. 1–10
DOI: 10.5604/20804075.1073042

Research Article

ENHANCE PERFORMANCE OF WEB PROXY CACHE CLUSTER USING CLOUD COMPUTING

Najat O. Alsaiari[1], Ayman G. Fayoumi[2]

[1] Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia, e-mail: nalsaiari@kau.edu.sa

[2] Department of Information System, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia, e-mail: afayoumi@kau.edu.sa

Received: 2013.09.18
Accepted: 2013.10.14
Published: 2013.12.06

ABSTRACT
Web caching is a crucial technology in Internet because it represents an effective means for reducing bandwidth demands, improving web server availability and reducing network latencies. However, Web cache cluster, which is a potent solution to enhance web cache system's capability, still, has limited capacity and cannot handle tremendous high workload. Maximizing resource utilization and system capability is a very important problem in Web cache cluster. This problem cannot be solved efficiently by merely using load balancing strategies. Thus, along with the advent of cloud computing, we can use cloud based proxies to achieve outstanding performance and higher resource efficiency, compared to traditional Web proxy cache clusters. In this paper, we propose an architecture for cloud based Web proxy cache cluster (CBWPCC) and test the effectiveness of the proposed architecture, compared with traditional one in term of response time ,resource utilization using CloudSim tool.

Keywords: Web proxy cache cluster, resource utilization, load balancing, cloud computing.

INTORDUCTION

Currently, the World Wide Web (WWW) is considered as the most successful application for providing simple access to a wide range of information and services. As a result, the amount of traffic over the Internet has experienced tremendous growth and most clients browsing the web and loading files through the Internet expect that they should obtain a fast service and one of the effective solutions of enhancing performance of the Internet services is using Web proxy cache. Proxy cache [1] is an important device to guarantee the quality of Web services, and it helps to lower the demand for bandwidth and improves request turnaround time by storing up the frequently referenced web objects in its local cache. However, in Web proxy system environments, load state changes frequently and we cannot predict the time or height of peak load. To overcome this issue, more resources added to guarantee system has enough capacity in order to handle peak load and maintain service quality level [7]. But, adding more resource means lower resource utilization and higher the system cost. Thus, a conflict occurs in between enhancing service quality and excelling efficiency of resources in web proxy cache clusters. In this case, Web proxy cache system often adopt load balancing strategies for obtaining a trade-off between those two issues of the system. However, performance analysis of various loading balance techniques used in Web cache proxy systems was well presented in [8]. The analysis indicates that both static and dynamic loading balance have a common issue related to resource provision i.e. whenever the load is low, it causes wastage of resources, whereas for being high the service quality deteriorates. In extreme overloaded situation (worst case), the Web proxy cache cluster drops services entirely.

Therefore, the load balancing strategy alone is unable to solve the conflict between improving resource utilization and enhancing quality of service. This conflict brings about the need to have a proxy system which is dynamically scalable i.e. having the capacity to adapting to the load that is needed (makes use of new nodes when there is an overloaded and removes nodes when there is light load). This is where dynamic scalability or *cloud bursting* is needed.

Cloud computing [9] is "a new paradigm, a type of parallel and distributed system, consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreement". And Cloud bursting is a new paradigm that applies an old concept to cloud computing, it represents the dynamic part of computing in the cloud. It describes the procedure in which new nodes are rented if needed and returned if not needed anymore. In proxy cache systems, cloud bursting means that if the system is overloaded for any reason, the load balancer automatically integrates new nodes from the cloud. Our work makes the following contribution:

- proposing an architecture for a Cloud based Web Proxy cache cluster system (CBWPCC),
- extending the capability of the cloud broker (as a user-level broker) to make the migration decision based on the application performance,
- evaluating the proposed architecture using cloudsim toolkit by considering different performance metrics.

RELATED WORKS

Significant number of work has been traced out so far on exploiting cloud computing for applications of data-intensive systems and scientific purposes. Systems for retrieving images based on contents are comprised of tasks related to high computations, as algorithms of those problems are of higher orders of complexities and volume of data are bigger. Therefore, to put up an image retrieval system, NIR needs to utilize cloud resources [3].

NIR is a kind of open source cloud which facilitated content based image retrieving systems. In BPEL workflow systems, virtual machines play significant roles. Tim Dorneman [4] employed it in Elastic Compute Cloud (Eci3 of Amazon) and hence, it was possible to provide new hosts in BPEL workflow systems and handle peak load situations there as well. Results obtained from experiments and assessments on applications of computationally intensive video analysis highlighted those solutions as feasible and competent. Several concerns were concentrated on "cloud bursting" compute model as throughput or response time of applications has been found improved through elastic allocation of cloud instances with local resources. Various job scheduling strategies have been considered by De Assuncao et al. and according to this concept computer nodes were integrated both at local site and the cloud [5].

There may stay possibilities of inclusion of time constraints with each job and jobs are vetted on submission following one of those job scheduling strategies. The decision is made by the system whether it needs the job to be executed on the cluster or the job is redirected to the cloud. Elastic sites (proposed by Marshall et al.) [6] maintain transparency in extending computational limitations of local clusters to the cloud. Computed decisions on EC2 for node (de)allocation are taken through middle-ware after assessing job queue of local clusters. In contrast, our proposed system makes calculated decisions on bursting requests to cloud proxies based on the requested documents type (miss/hit) and their average service time. Closely related research [7] considers integrating local proxy cluster in cloud environment.

In our study we present a fine grain architecture of cloud based proxy cache cluster and consider the response time, resource utilization and the cost of this integration.

THE PROPOSED SYSTEM

Figure 1 shows high level architecture for cloud based proxy system and its connections to the corresponding components that together represent an interoperable solution for establishing a cloud based web proxy cache cluster. There are basically four main entities involved:

1) Cloud Information Service Registry (CISRegistry).
2) The broker.
3) Virtual Machines.
4) Physical Machines.

Cloud Information Service Registry (CISRegistry)

Public datacenters register their information to the CISRegistry. It provides utility computing service to Cloud users/Brokers. The Broker in turn use the utility computing service provided by the CISRegistry to become SaaS providers and provide services to their end users.

The broker

It is the key player in this model would be able to make use of the results to provide means for disturb and burst requests and it consists of two components:

- **Load manager** – has three main functions: (1) It receives the user's requests and processes them on FCFS basis, distributes the load among its local/public proxies under round robin policy and queuing the tasks when there is a available capacity for them in the infrastructure. (2) It takes the responsibility of creating virtual machine image and their initiation on selected physical hosts. (3) It also keeps track of the execution progress of service requests i.e. the performance is periodically monitored to identify exceeding in the predefined threshold and thus send a notification trigger to Resource Allocator for taking appropriate action.
- **Resource Allocator** – its responsibility is to handle the state of an overloaded system. The allocator queries the CISRegistry for the datacenters information when utilization goes over the predefined threshold. The CISRegistry in turn responds by sending a list of available datacenters to the Resource Allocator in order to start bursting the requests to the selected public datacenter.

Virtual Machines

Multiple VMs can be started and stopped dynamically to meet accepted service requests. In addition, multiple VMs can concurrently run applications based on different operating system environments on a single physical machine since every VM is completely isolated from one another on the same physical machine. However, we assume that cache application is composed of one virtual machine that housed in local cluster (private datacenter), which offer mechanisms for dynamic scaling of server capacity.

Physical Machines

The cluster comprises multiple proxy servers that provide resources to meet service demands. CBWPCC has $n + m$ back ends (including n local nodes and m cloud nodes, where $n > 0$ and $m >= 0$). Local nodes remains as part of the CBWPCC as

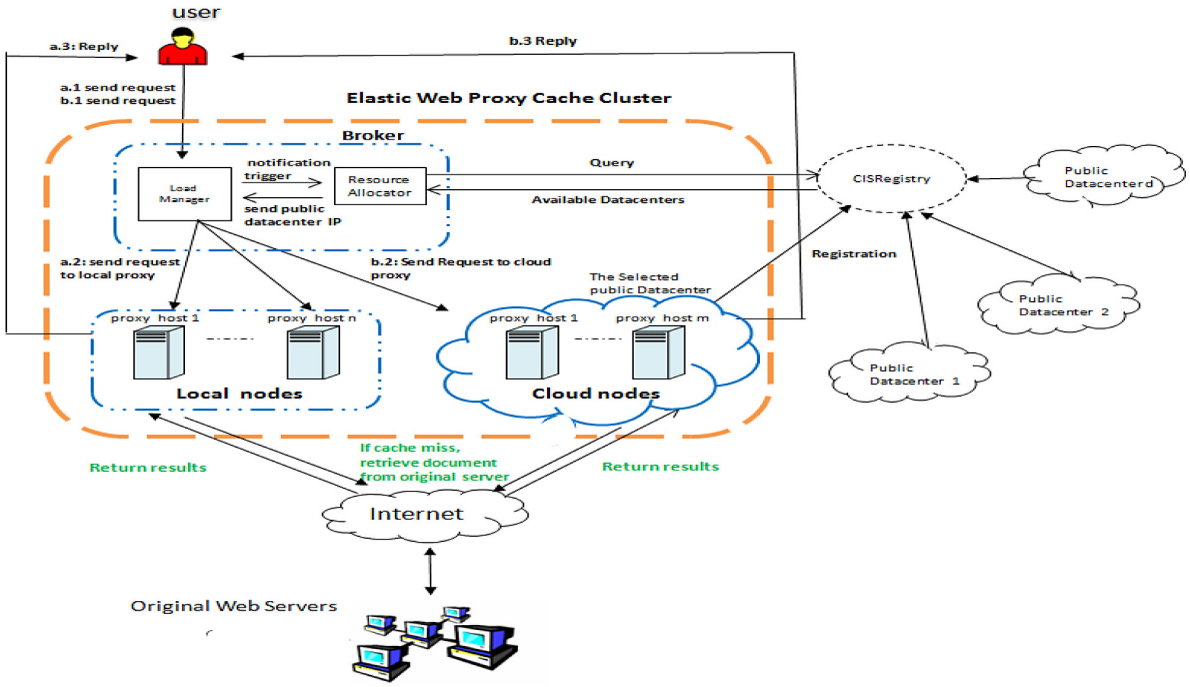

Fig. 1. Cloud based Proxy Web Cache Cluster Architecture

long as they work normally. Cloud nodes are constructed using cloud resources and they act same as local nodes functionally. The only difference is that the cloud node number changes with load status and the amount of available resources. In perspective of hybrid supply mode, CBWPP has the ability to use both local node and cloud node at the same time. Our system uses cloud resources for insuring of service quality and system performance in case of being overloaded and for providing reliability of CBWPCC, local nodes are used here. The proxies (local/cloud) retrieve and process the assigned requests. If the data is stored (i.e. hit cache), the requested document will be retrieved from the local proxy. If the requested document is not cached (i.e. misses cache), the proxy will redirect the request to the original Web server.

To ensure fast deployment of cache application in public cloud, we use fully configure image deployment method to construct cloud nodes for our system [13]. In this deployment method, the cache application is already installed and configured to create virtual machine in clouds.

PERFORMANCE METRICS

A number of metrics have been commonly used for evaluation of performance of the web cache. These metrics are related to cache efficiencies. Hit Rate (HR) which is the ratio of documents attained from usages of cashing mechanism against total documents required. Byte hit rate can be evaluated from the ratio of the number of bytes fetched from cache and the number of bytes that have been accessed. There are other two measures, named as user response time and bandwidth utilization, which can mitigate volume of bandwidth consumption. Besides, CPU and I/O system utilization , the fractional part of available CPU cycles or latency of disk and object retrieval got considerable attention to end users as other measures of the same objective.

In this paper, the response time is chosen for evaluation and assessment of the performance of web cache cluster. This performance metric is vastly used in this arena. However, from response time perspective, the existing web cache system may be considered in an ideal condition, as long as it can decrease the user's response time significantly. Still, because of the cache application's characteristic, some requests should be redirected to the original server since their requested documents are not cached in the local disk of the proxy. These requests are termed as 'miss requests' whose response times are influenced with some attributes such as network conditions and process capacity of the original server. During the time of the congestion in the network or if the original server is overloaded, the response time of 'miss request' will be raised distinctly even if the load in CBWPCC is low. Thus, response time cannot be used as a direct performance index. Instead, we adopt a relative performance index *R* from [7]. it is the ratio of average service time of hit requests to original server average response time of miss requests in a given period of time.

$$R = \frac{\text{AvgHitTime}}{\text{AvgMissTime}} \tag{1}$$

$$\text{AvgHitTime} = \frac{\sum_1^{\text{i=number of hits}} \text{service time of hit request i}}{\text{i}} \tag{2}$$

$$\text{AvgMissTime} = \frac{\sum_1^{\text{j=number of miss}} \text{response time of miss request j}}{\text{j}} \tag{3}$$

While using *R*, a threshold range of values need to be set such that whenever the value of *R* goes beyond the threshold value, a new node is added to the cache cluster. In other cases, cloud nodes are removed from the cache cluster. However, setting the threshold value is an important issue and it should be decided at an optimum range for two reasons: the performance of the system should be maintained to an acceptance level and frequency of node number's alterations should be as low as possible to avoid performance jitter.

Assume that, a threshold range is set as (*t1*, *t2*) where *t1* is the lowest range value and *t2* is the highest range value of the threshold value ranges. Now, if $R < t1$, the system will drop or release cloud resources if there are cloud nodes. If $R > t2$, the system will acquire cloud resources and build cloud nodes to increase proxy cache system's capacity. If *R* retains within this range, existing cloud nodes will be kept in rent and stream of incoming requests will be disturbed amongst local and cloud nodes. Figure 2 shows an activity diagram of our system.

COMMUNICATION AMONG ENTITIES

We modified the original CloudSim communication flow to another one shown in Figure 3. First, each public cloud datacenter registers itself with the CISRegistry (Cloud Information Service).

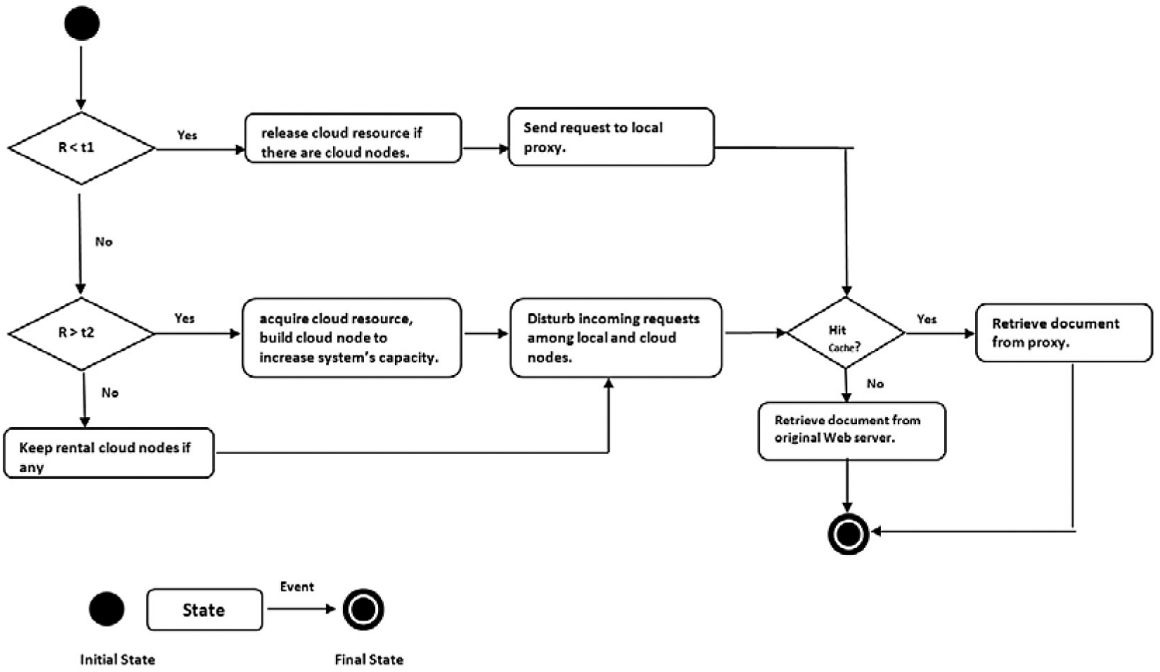

Fig. 2. Activity diagram

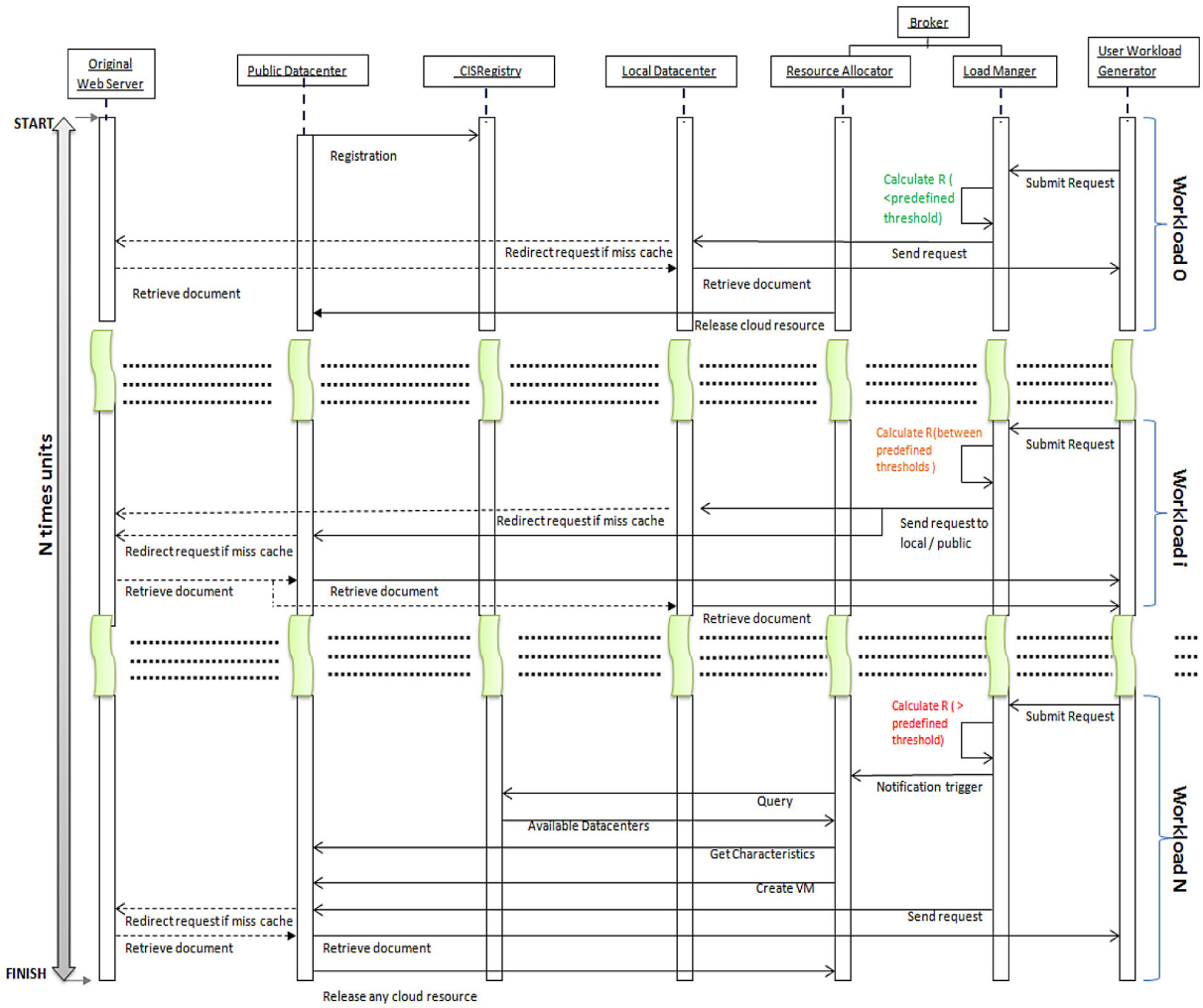

Fig. 3. Communication flow

CISRegistry provides database level match-making services for mapping user requests to suitable Cloud providers. The broker queries the CIS-Registry for a list of cloud datacenters which offer less latency and response time services. In case the match occurs and since we use an image fully install cluster the broker only needs to send requests directly to the selected Cloud datacenter. The simulation ends after this process has been completed in the original flow. Therefore, we added a new entity, called User Workload Generator, to periodically impose load on the system for *N* steps/periods (time unit = second). As can be seen from Figure 3, the performance index *R* is calculated at each step/period. The workload is defined as the number of requests from end users.

PERFORMCNE EVALUATION

This section describes results from a detailed evaluation study we performed. Particularly, we evaluated the feasibility and performance of CBWPCC.

Data Collection

In order to verify performance of the proposed architecture, we have implemented a trace-driven simulation of Web proxy cache cluster. We select squid access log of NLANR to be the trace in simulation compare CBWPCC with traditional Web proxy cache clusters (TWPCC) which have fixed node number and do not have the elasticity feature. The traces used for this study are downloaded from the NLANR website: ftp://ircache.nlanr.net/Traces. These one-day log traces were collected from the proxy locations in Palo Alto, CA. However, there are many possible responses that a Web proxy can provide to a client request [10], but we only considered GET requests such that the cache returned a "200" or "206" response code to its clients, Table 1.

Experimental Setup

This research utilizes a simulator environment where the CloudSim-2.1.1 is used as a framework [11] along with Java version 7 [12] and other necessary execution setup. We have set the threshold value *R* too (0.9, 1.2) and sample period to 120 seconds. The number for local nodes of CBW-PCC is fixed to 8.

In this simulation setup, a broker received the user's requests and processed (queue, execute) them on a FCFS basis. To evaluate the effectiveness of the proposed system in response time and resource utilization (CPU, memory and disk), two test scenarios were simulated: in the first scenario, all the workload was processed locally within the local proxy cluster (private cloud). In the second scenario, the workload (requests) could be migrated to public cloud in case the workload in local resources exceeds a predefined threshold. In other words, the second scenario simulated a CloudBurst by integrating the local private cloud with public cloud for handing peak in service demands. We assume that the broker migrate requests to a public cloud with minimum latency and response time.

In Figure 4, a comparison of performance result is depicted between the CBTWPCC and TWPCC, where number of fixed nodes were 8 and16. In this graph, from the initial point of 1000th period, the average response time curve of CBTWPCC is similar with that 8-node TWPCC. When the load of the system is enlarged, we can see that 8-nodes TWPCC cannot afford enough resources and its response time increased notably. While CBWPCC could get more resources from cloud and capacity of the system could be expanded dynamically. Therefore, the average response time of our system is maintained at a relatively lower level.

In case of the 16-node TWPCC, the beginning point at 1000th period, the response time of our system got raised as our system initially has 8 nodes only. After the beginning point, cloud

Table 1. HTTP Response Codes from [2]

Response Code	Description
TCP_HIT/200	valid document was made available to the client directly from the proxy cache
TCP_HIT/206	a partial transfer of the document directly from the proxy to the client
TCP_MISS/200	valid document was made available to the client by retrieving the document from another proxy cache or from the originating server
TCP_MISS/206	a partial transfer of the document from the originating server

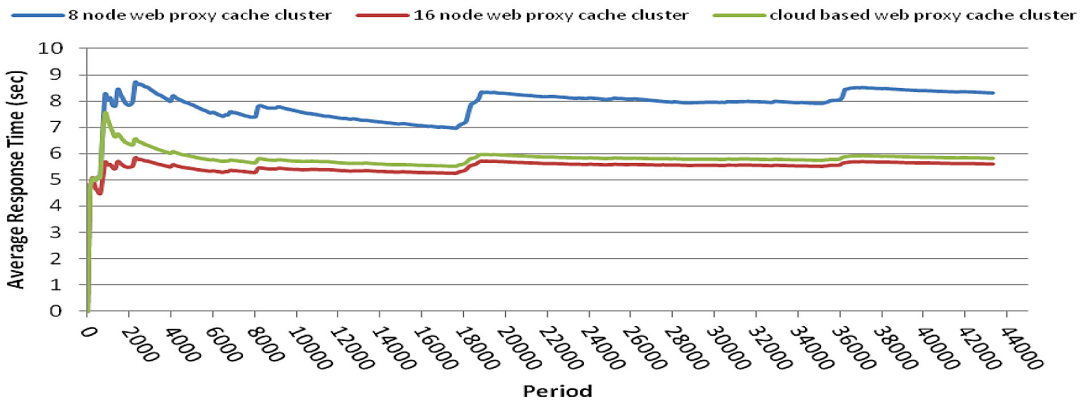

Fig. 4. Performance comparison of proxy Web cache cluster

nodes are added into the system dynamically and at the 18 000th period, the average response time of the system is going almost the same with that of 16-node TWPCC. It is noticeable that average response time curve of our system is slightly higher than 16-node TWPCC because of the latency incurred for those requests that are sent to public cloud.

In Figure 5, a comparison of CPU utilization is graphed depicting CBWPCC comparing with TWPCCs. CPU utilization of 8-nodes TWPCC is leading all, and all three curves are maintaining approximately a constant distance to each other. The curve of the TWPCC originated with the same point of that of the proposed system. After starting from the origin, curves of all three did not raise much with progression of the period. But from 0 to 1000 all of them raised sharply (with exponential rise). All 8-nodes TWPCC and CBWPCC followed the same curve line rising sharply up to 65% CPU utilization at 1000 period. After that period, CPU utilization of 8-node WPCC goes little higher, as the curve raised very little, due to the cloud elasticity feature of our system cause the total number of requests processed locally less than total number of requests processed in 8-node TWPCC in a given period of time. On the other hand, the curve of 16-nodes TWPCC is significantly lower than that of other two. Even up to 44 000 period of time, utilization of CPU did not go above 55%. While it reached 90% and 85% for 8-nodes TWPCC and CBWPCC respectively. CPU utilization of CBWPCC is significantly higher than 16-node TWPCC, since our system has only 8-fixed local nodes and thus they suffer CPU overhead higher than 16-nodes TWPCC which has 16-fixed nodes.

Functionally CPU and RAM are co-related to each other in a cache proxy systems. As CPU utilization as resource is lower in CBWPCC to 8-nodes TWPCC, the scenario is the same for utilization of memory as well, Figure 6. All three curves tend to be unchanged after 1000 period of time. However, sharp rising of edges were seen from the origin to 1000 period. As all requests are processed locally in 8-node TWPCC, it is getting higher overhead from requests that are consuming memory resources. While our CBWPCC has the ability to migrate part of incoming requests into cloud when *R* exceeds the predefined threshold, its local memory resources face little overhead because the number of requests processed locally

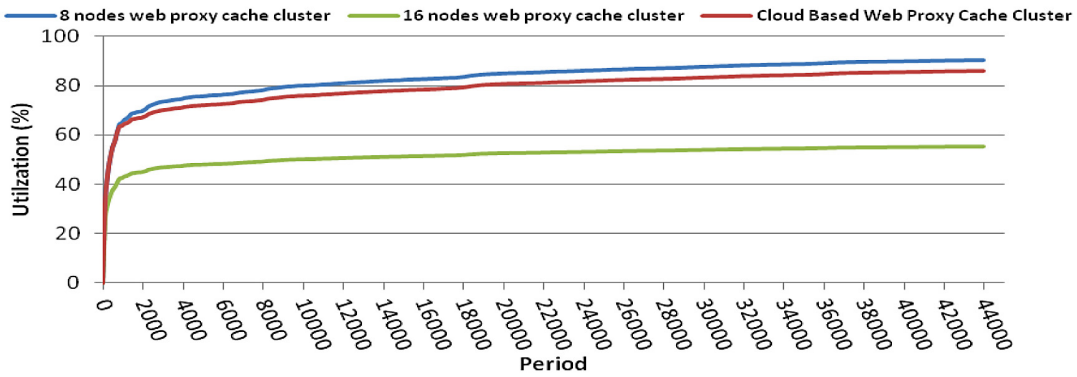

Fig. 5. CPU utilization

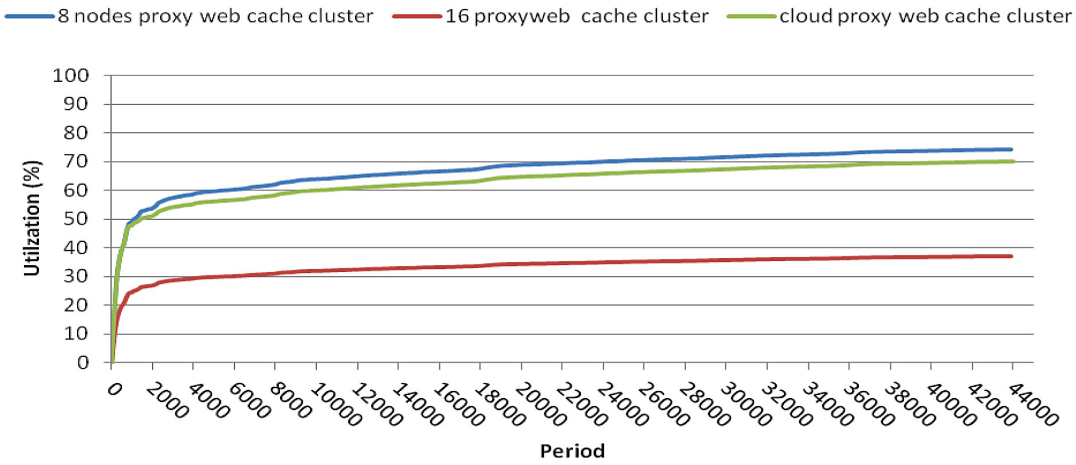

Fig. 6. RAM utilization

become approximately a half of total number of requests processed in 8-nodes TWPCC. In other hand and compared to 16-node TWPCC, our system has higher significant memory overhead since it has only 8 fixed local nodes with limited memory spaces.

In CBWPCC, we noticed that CPU and memory usage increases slightly to 65% and 50% respectively before acquiring new nodes from the cloud and both CPU and memory resources did not reach a critical overhead level. This is because our system is designed not to trigger on CPU/memory metrics but on another metric *R* crossing a threshold value.

In Figure 7, the utilization of I/O is much more effective in CBWPCC than in 8- and 16-node TWPCC. Though all three curves originated from the same point, with the gradual growth of workload rising, the curve of CBWPCC is higher than that of the two traditional web proxy cache clusters. Thus, our proposed system can handle more contents of bytes in a given period of time since relieving workload on our local nodes cluster by bursting some requests to the cloud helps in enhance the disk I/O utilization of CBWPCC by 4.35%.

Result of VM cost with cost of data transfer and total cost of adding cloud nodes is depicted in a bar chart in Figure 8. The VM cost has four components:

- per processing cost,
- per unit cost of memory,
- per unit cost of storage and
- per unit cost of used bandwidth.

Cost per memory and cost per storages are incurred during creation of virtual machine. Cost per bandwidth occurs during transfer of data. Other costs, processing costs, are relevant to uses of processing of resources. The pricing policy was adopted from Amazon's business model of small instances ($0.06 per Standard On-Demand instance per hour) which means that cost per instances are charged hourly [14].

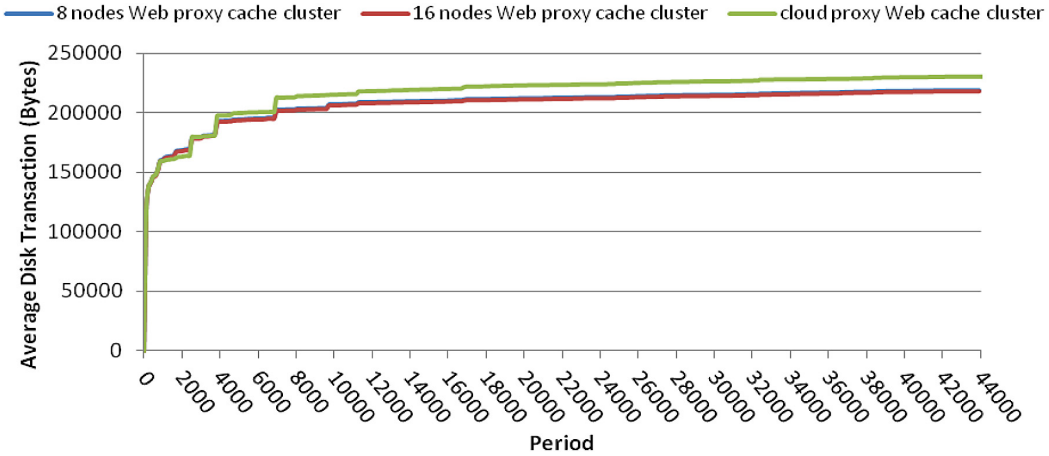

Fig. 7. I/O utilization

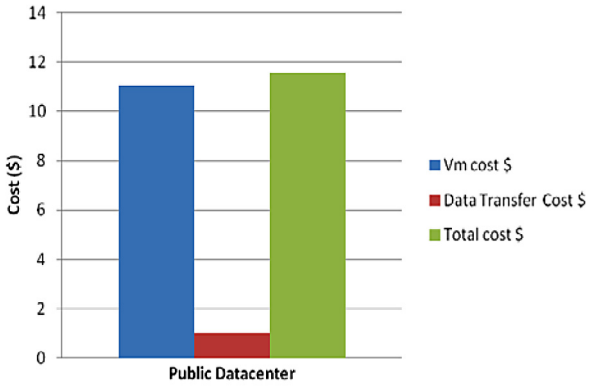

Fig. 8. Cost of bursting cache requests to public cloud

CONCLUSION

Load balancing strategies are adopted generally by traditional web proxy cache clusters to get a tradeoff in enhancing service qualities and resource efficiencies. Though, the solution seems to be reasonable, it will result in resource wasting when the load of the system is low. When the load is high, the service quality will be poor and it will be dropped in its worst case situation (when the system is exceedingly overloaded). To overcome this issue, this paper proposed an architecture for a cloud based Web proxy cache cluster. By using on demand cloud resources, the system to expands its capacity when the load goes beyond its local capacity and released them when the load gets down.

We evaluated the effectiveness of the proposed architecture comparing with traditional one which have fixed node number, in term of response time ,resource utilization. Results of the experiments shows that Cloud based Web proxy cache cluster performs better in obtaining of higher resource efficiency and lowering of system cost. And since we deploy the cache application in cloud environment, this may bring security issues. Therefore, our future direction will focus on presenting a solution to avoid the influence of cloud security for the proposed cloud based cache proxy system.

Acknowledgment

This paper contains the results and findings of a research project that is funded by King Abdulaziz City for Science and Technology (KACST), Grant No: T-T-12-0938.

REFERENCES

1. Zeng D., Wang F., Liu M. Efficient web content delivery using proxy caching techniques. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, Aug. 2004, 34(3): 270-280.
2. Mahanti A., Williamson C. Web proxy workload characterization. Technical Report, Department of Computer Science, University of Saskatchewan, February 1999, http://www.cs.usask.ca/faculty/carey/ papers/workloadstudy.ps
3. Yang Z., Kamata S., Ahrary A. NIR: Content based image retrieval on cloud computing. [In:] Proc. IEEE International Conference on Intelligent Computing and Intelligent Systems, 2009, IEEE Computer Society, Shanghai, China, 2009, p. 556-559.
4. Dornemann T., Juhnke E., Freisleben B. On-Demand Resource Provisioning for BPEL Workflows Using Amazon's Elastic Compute Cloud. [In:] Proc. of the 9th IEEEIACM International Symposium on Cluster Computing and the Grid, IEEE, Shanghai, China, 2009, p. 140-147.
5. Assuncao M., Costanzo A., Buyya R. Evaluating the Cost- Benefit of Using Cloud Computing to Extend the Capacity of Clusters. [In:] Proc. of High Performance Distributed Computing (HPDC), June 2009, p. 141-150.
6. Marshall P., Keahey K., Freeman T. Elastic Site: Using Clouds to Elastically Extend Site Resources. [In:] Proc. of Conference on Cluster, Cloud, and Grid Computing (CCGRID), May 2010.
7. Duan Z.; Gu Z. EWPCC: An elastic Web proxy cache cluster basing on cloud computing. [In:] Proc. of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), July 2010, 1(9-11): 85-88.
8. Alsaiari N., Fayoumi A. Load Balancing Techniques for Web Proxy Cache Clusters. International Journal of Advanced Research in Computer Science (IJARCS), Sep-Oct 2012, 3(5).

9. Buyya R., Yeo C.S., Venugopal S. Market oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities. [In:] Proc. of the 10th IEEE International Conference on Advanced Learning Technologies, July 2010.
10. Rodriguez P., Spanner C., Biersack E.W. Analysis of Web caching architectures: hierarchical and distributed caching. Networking, IEEE/ACM Transactions on, Aug 2001, 9(4): 404-418.
11. CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, 2011 (available from: http://www.cloudbus.org/cloudsim/)
12. Java software version 7 downloaded from: http://java.com/en/download/index.jsp (August 2013).
13. Krsul I., Ganguly A., Zhang J. VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing. [In:] Proc. of the ACM/IEEE Conference on Supercomputing, IEEE Computer Society, Pittsburg, PA, USA, 2004: 1-12.
14. Amazon EC2 Pricing. Pay as You Go for Cloud Computing Services. Amazon EC2, n.d. Web. 14 Sept. 2013. <http://aws.amazon.com/ec2/pricing/>