International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 2– No.1, May 2012 – www.ijais.org

Learning Framework with Load Balancing and Fault Tolerance using Cache-based Architecture

K. Zafar, Q. S. Haq, J. Kansi, A. Khan, S. Zafar, Z. Halim
National University of Computer & Emerging Sciences
Islamabad

ABSTRACT
Most of the universities around the world are offering online distance learning programs. A framework is required that can handle huge amount of network traffic at the servers and also cater for the autonomous, robust, intelligent load balancing and fault tolerance. In order to handle large number of transactions at the server, high performance server architecture is required. The proposed framework is tested using a virtual online university that can handle huge amount of user requests. With the availability of high bandwidth and low cost hardware, the optimum performance can be achieved easily. The frame work is tested under heavy network traffic and proved to robust, scalable and reliable approach.

Keywords
Distance learning, cache, fault tolerance

1. INTRODUCTION
The handling of large number of clients on a web server has been an area of interest since many years. For the requirement of an online university with hundreds of users as clients, a high performance transaction processing server is required. This frame work will have a reconfiguration module that will solve the traffic congestion problem at the database server. The users for such a system can be categorized as students, teachers and administrator. The role of the administrator can be ignored as there server hits are the least. The teachers and the students are the main users that can cause network congestion. This system provides an intelligent teacher-student interface with the capabilities of chat, video conferencing, interactive tutorials and robust examination module. The handling of video conference with hundred of users simultaneously along with real time testing module are the main focus of this research as discussed in [1-3]. The students are the main concern as they are the main clients that will be located anywhere in the world and will be using the system through high speed internet. The server will be busy most of the time for handling online exams and video sessions for the lectures.

2. TRAFFIC CONGESTION AND PERFORMANCE
The first client of the system architecture as shown in Fig.1 is the administrator. The interaction with the server is not frequent. The role of an administrator can be ignored as far as traffic congestion and performance is concerned. The reason is the number of accesses they would make at any time. Since there role is only to provide facilities like adding new users, starting new courses and performing other administrative jobs.

The second type of client is the teacher. The number of teachers is relatively higher then administrative staff, but at the same time it is relatively small as compared to students count. The main role of the teacher is to deliver a video lecture and creating exams. The exams are checked and evaluated by the system autonomously. The teacher can create exam, upload lecture slides, schedule the lecture and exam sessions. For one thousand students there will be around 50 teachers if each teacher is teaching on average 3 courses then each teacher will create 30 exams in one semester (six months). And hence fifty teachers will create 50*30 = 1500 exams at maximum. This figure also shows number of transactions that will be made in database. This is not a huge number of transactions in 6*30=180 days. Hence this can be neglected as well.

The third clients are the students that are the main focus in the development of this framework. They require the maximum number of hits to the server.

The first type of transaction that a student will make to the server is lectures. This is something that will not affect the performance of the server at any time. As discussed above for teacher, student's situation will be quite same. The main role of server and database will be to authenticate the client and then teacher and student direct interaction will start. After this, main problem will be bandwidth required for video streaming at student's end. The next and most critical transaction type is exam and quizzes. We know that there are three exams (including midterms) on average and around ten quizzes on average in one semester. Suppose there are 50 students enrolled in one course (actual number might be higher than this because of online university). If there are twenty lectures going on simultaneously, and all of them are taking a quiz, then there will be 20*50 = 1000 students accessing the server at same time and all of them accessing database as exams and quizzes will be stored in main database. This will create a severe problem for the server. The response will be extremely slow and it might even crash due to high load. This is just one example, actual problem might be even worse.

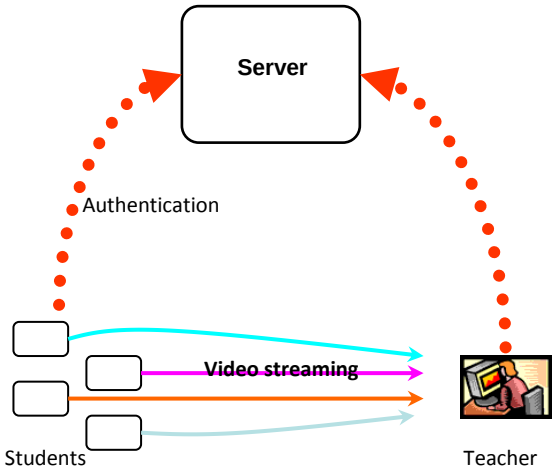

Figure.1. System architecture

Students might also access server for checking their grades, submitting assignments, viewing notice board, registering in courses, chatting, messages, etc. Number of these transactions will also be high and affect the performance of the server.

One thing that must be noticed here that main problem arises when exams or quizzes are going on. It must also be noticed that at such a time most of transactions to the database will be for retrieving exam data/questions from database and also that since exam will be taken by all students of a course, all these queries will be same and would run again and again. This research will use this logic for improving efficiency of the server as discussed in literature review from the papers [4-7].

3. LITERATURE REVIEW

3.1 Tech-Brief Extreme Networks

The architecture presented in this paper [3] as shown in Fig.2 has special requirements for hardware. The integrated server approach allows high speed data transfer and disaster recovery. The system performs efficiently on local network by utilizing potentials of connecting medium. Such architecture can't be used in web based systems, as in our case the requirement is load balancing and fault tolerance.

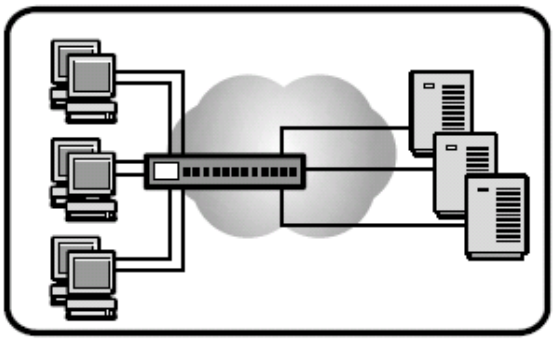

Figure.2. Tech-Brief extreme network

3.2 Web Distribution Systems: Caching and Database Replication

This paper [4] gives idea of using Multicast based protocols to perform caching and replication. The problem is that neither can be used in isolation. Increasingly, caching and replication systems are being setup to reduce network latency in the backbone. Among caching products proxy server caches have a cost advantage over Appliances, even though the later display better performance. The concept of cluster based caching is not very useful for a web server whose core requirement is load balancing and fault tolerance. Especially in case of a database for an online university, maintaining such a cache would require extra effort in terms of synchronization and constant updates.

3.3 Web Server Accelerator Design

In paper [2], authors presented a technique for improving the performance given by this paper is to cache data at the site so that frequently requested pages are served from a cache which has significantly less overhead than a Web server. This may be desirable for several reasons as shown in Fig. 3:

- Multiple nodes provide more cache memory. Web server accelerators have to be extremely fast.
- Multiple processors provide higher throughputs than a single node.
- Multiple processors functioning as accelerators can offer high availability.

The architecture presented in the paper is good for high performance web site connectivity, but this design is not suitable for dynamic web servers. As in our case load balancing is a problem at back end, not an issue of storing contents of web pages.

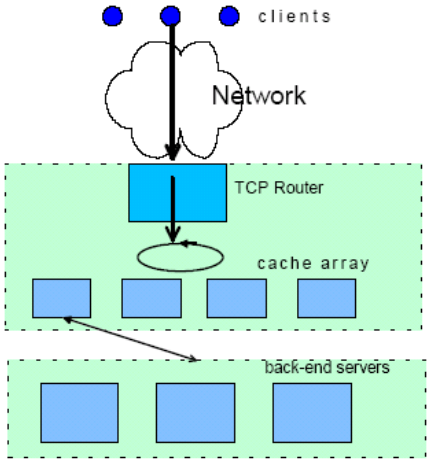

Figure.3. Web server accelerator

3.4 Dynamic load balancing of web servers in geographically distributed heterogeneous environment

This paper [1], presents architecture for load balancing and high speed data recovery in a web server. The problem with this architecture is that it is not suitable for our domain problem. This architecture improves efficiency by utilizing potentials of scheduling algorithms to distribute load among different servers that are geographically apart.

4. ARCHITECTURES ANALYSIS

To improve the performance of the server in situations as discussed above, different design structures of the server were developed, analyzed and tested to find the best possible design for the server that will address all issues of traffic congestion and for increasing performance and efficiency of the server.

4.1. Single server

This is the simplest and easiest design to implement. It involves a single server where all users will login. This design still has the above mentioned issues. The database will reside on the server and IIS will run on the same machine as well. The number of users will affect performance of this server.

4.2. Single server with cache

To improve the performance of the single server, cache can be incorporated. As discussed in previous topic, most of queries at the time of examination would be same. If the results of query are in cache, the user request can be handled more than once. This will save a lot of time that was previously wasted by accessing database again and again for same query. This design still does not solve problem of high number of users. One thing must be kept in mind that cache is used only for search queries, not for insert or update queries. If we want to use cache for all kind of database queries then it requires synchronization between database and the cache, which is very costly as number of update and insert queries is high and almost equivalent to search queries. The benefit of cache will fade out due to constant synchronization all the time.

4.3. Multiple servers with database replication

The key idea is to have more than one server to handle clients at one time and each server has its own database which is replica of master database as shown in Fig. 4.

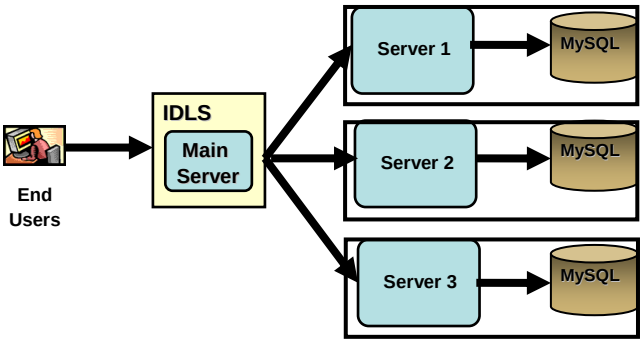

Figure.4. Multiple servers with database replication

The benefits for this design are:

1. Clients will divide and hence lesser traffic on any one server
2. Each server will have its own database and hence lesser time will be spent on database query transactions

Although such a design has many benefits, but at the same time it also has huge problems associated with it, solving these problems create more performance and efficiency problems.

The replication of a database is not an easy task, especially when there is one master and multiple slave replicas. If there is a change in one database it must be changed on all databases simultaneously.

This effectively reduces efficiency of the whole system. And since there will be changes almost all the time, the time taken by these changes will be much higher than the time saved by introducing multiple servers. Hence this is not a good design and solution to our problem.

4.4. Multiple servers with database replication and cache

In single server with cache design, the introduction of a cache module increases performance of the server by reducing the time wasted on running same queries on database again and again. The introduction of cache in the design with multiple servers with database replication and cache as shown in Fig.5, the performance of the server increases effectively, as most of the queries running in the system at the time of examination are almost same, especially in case of students with the same course.

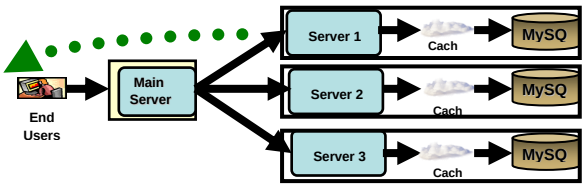

Figure.5. Multiple servers with database replication and cache

This design still does not solve problem of database replication. One thing must be kept in mind that cache is used only for search queries, not for insert or update queries. If we want to use cache for all kind of database queries then we will need synchronization between database and this cache, which is very costly as number of update and insert queries is as high as search queries. And the benefit of cache will fade out due to constant synchronization all the time. As only cache is introduced and this will only enhance the search queries time for database transaction, not insert or update queries.

4.5. Multiple servers with single database and cache at two levels

This is the final and most efficient design for our problem. We will use all the benefits of multiple servers and cache to solve problems of traffic congestion and low performance. At same time we will not use database replication idea and hence will not need to solve problems associated with database replication.

Two level caches have been used to reduce the database transaction time. The details as shown in Fig.6 are discussed below.

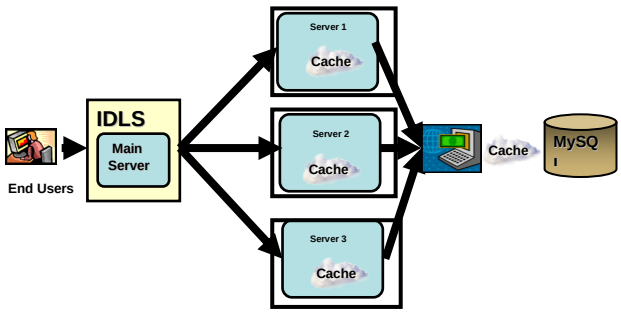

Figure.6. Multiple servers with single database and cache at two levels

As discussed earlier the two main issues to resolve are client requests load and data load. The first problem is handled using multiple servers instead of just stand alone web server and the second issue is resolved using two level cache and single database server.

4.5.1 Multiple servers

The key idea is to have more than one server to handle clients at one time. When a request comes from user (connection to server), this is directly forwarded to main server. Main server has detail of all the servers running at the back end and uses round robin algorithm to shoes among these servers running at back end. It must be noticed that the job of this main server is only to route the client requests to server that has minimum number of clients connected to it. There will be added benefit for having such kind of design i.e. clients will be divided among all sub servers and hence lesser traffic on any one server at a time. When users are connected to sub servers, user can use the online facilities and do any kind of transactions. Since there are multiple servers running here, there won't be any problem of traffic congestion. And we can add as much servers as we want to tackle growing number of students of online university. Adding a new server is very easy, we only need to inform main server the address of new server and the main server will start automatically routing clients to this new server.

4.5.2 Database server

The main idea for having a separate database server is to get rid of problems of database replication and still have multiple servers. This separate server will have web service running on it, which will provide interface for sub servers to use the main database. Now all the servers will request this server for any data they need from database. Although all these servers are on same network and there won't be issue of machine-to-machine time laps but there will still be some time consumption between request and reply mechanism of web service and surely network will have to be fast enough to support such kind of mechanism. To solve this problem the paper proposes the idea of multiple level caches that will reduce database time consumption by storing query results at two levels.

4.5.3 Two level Cache

What happens when server needs to fetch data from database? The answer is not simple; this query has to go through many levels before actually reaching database. At first step the server will check that if data is available in its cache or not, if it is available in the cache then it is simply returned to the HTTP page, otherwise this query is forwarded to a different server which is specially designed for database transactions.

Database server has a cache for further improving database retrieval time. As discussed earlier most of queries at time of exams will be same so if same queries are coming from different servers then this cache will help in saving time of running same query again and again

5. IMPLEMENTATION

The architecture presented in the paper was implemented using Dot Net Tools. A web application "router" was created to provide interface for clients. All clients first connect to this application; the main job of router was to select a sub server

which has lowest number of clients connected to it, after this router simply forward the request to sub server. Web services are used to connect router and sub servers to database server. The host database server receive queries as strings and return results in form of dataset, which is serialize able in XML. Third part of this architecture is sub server, which are actually replicas of same server. Clients at each server can perform any kind of operation which they are authorized to.

5.1 Comparison of architectures

The architectures are implemented and analyzed to give clear picture of optimized results as shown in Fig. 7. A graph below shows a small comparison of different scenarios and respective behaviors of architectures.

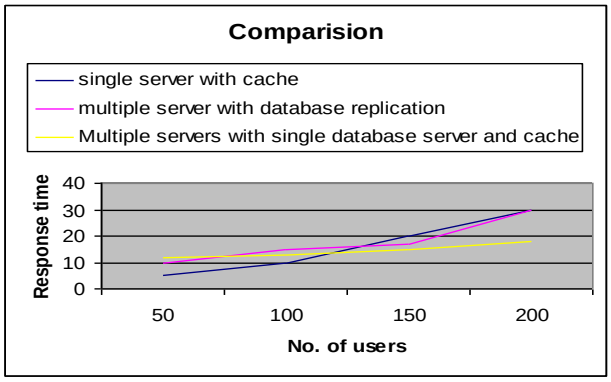

Figure.7. Comparison of architectures

5.2 Autonomous Traffic Monitor

The architecture has many components, which can increase the failure point as compared to single server. Since it is possible that traffic congestion might occur due to extra load on one server, so there must be some mechanism to actively monitor the process at all times. To solve this problem, a module as shown in Fig.8 is designed to provide facilities of monitoring the architecture. The module is independent and takes decisions on its own, but human interference is also allowed for oracle involvement. If clients on one server exceed a threshold set by administrator, then that server is temporarily shutdown. Shutdown does not mean machine power off, but it means that no more clients will be routed to this server until the load is balanced among all the sub servers. After the load is balanced, this server is allowed to take requests again.

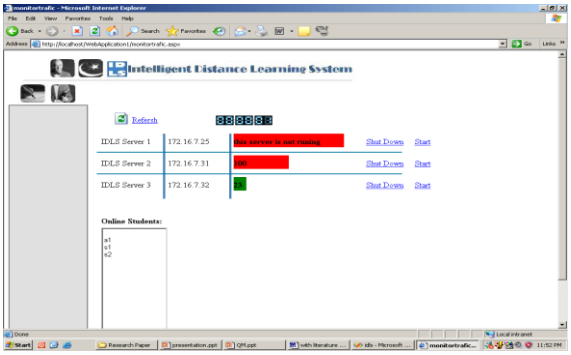

Figure.8. Overloading of server

6. CONCLUSION

Traffic congestion is a problem that is always faced when number of clients accessing a web server increases beyond limit. In case of an online university, where there are special problems and needs specific solution. The presented framework design given in this paper has been formulated after careful analysis of the problem and also looking at different aspects of its implementation. The presented framework is implemented and tested. It is proved to be robust, scalable and reliable.

7. REFERENCES

[1] M. Yu, P. S. Cardellini, V. D. Sistemistica. 1998. Dynamic load balancing in geographically distributed heterogeneous Web servers. In proceedings of 18th International Conference on Distributed Computing Systems, p.295–302.

[2] Cardellini, V. Colajanni, M. Yu. 1999. Dynamic load balancing on web-server systems. In Proceedings of IEEE International Conference on Internet Computing.

[3] Dias, D. M. Kish, W. Wukherjee, R. Tewari. 1996. A scalable and highly available web server. In Proceedings of International Conference on Technologies for the Information Superhighway, Compcon.

[4] D. Skeen. 1982. A Quorum-Based Commit Protocol. In Proceedings of the workshop on Distributed Data Management and Computer Network, p.69 - 80.

[5] R. Thomas. 1999. A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases. In ACM Transactions on Database Systems.

[6] M. Harchol-Balter, B. Schroeder, N. Bansal, M. Agrawal. 2001. Size-based scheduling to improve web performance. ACM, New York, NY, USA.

[7] K. Park, V. S. Pai. 2006. Scale and Performance in the Co-Blitz Large-File Distribution Service. In Proceedings of the 3rd Symposium on Networked Systems Design and Implementation, San Jose, CA, NSDI.