Scaling Technique for Web Based Management Systems in Bioinformatics

Syed M. Ahsan, Amjad Farooq, M. Shahbaz, M. Junaid Arshad, M. Aslam

Department of Computer Science and Engineering, UET, Lahore, Pakistan
ahsancs@hotmail.com

Abstract: With the enormous growth in web based management solutions the number of transactions increases proportionally, as system grows the number of requests increase. With the increase in number of request there introduce scaling concept. There are number of scaling techniques available but all these techniques focus on scaling the server by distribute the load on different servers. Many techniques give solutions to reduce the number of request and load balancing. But all the existing techniques are limited as the number of transactions increases exponentially. As a remedy we introduce new technique, which helps give a scalable solution without server maintenance cost. In this technique all users actions stored in the local java script model which then will saved in the database after some point so that to reduce the traffic. Objective is to make some intelligent client who stores all transaction at the client side and send only one request to server for all transactions.

[Syed Ahsan, Amjad Farooq, M. Shahbaz, M. Junaid Arshad, M. Aslam. **Scaling Technique for Web Based Management Systems**. Life Science Journal. 2012;9(3):1-5] (ISSN:1097-8135). http://www.lifesciencesite.com. 1

Keywords: Techniques; Management System; Web Based; Solutions

1. Introduction

The main issue with web applications is inability to plan and predict the number of users that will be accessing this application. Sometimes this amount of users generate peak load in some special events or days. In such condition application remain to continue and work properly without any impact. And the response time must be average. So in the worst case application may chock or server crashes because of the amount of users that access the system. So the solutions would be such that it handles such worst condition and handle the dynamic behavior. Many techniques offer some dynamic behavior of the server, which distribute its load and give the average response time. In all techniques they might need some minimum number of servers but in the worst condition the number of servers might increase to infinite so it would increase the maintenance of server and cost of each instance. Many scaling techniques exists which helps to improve the server response and load on server, Like database Sharding which shards the data with respect to region using different shard key, Replication of servers, but all the available techniques have high cost of maintenance and hardware [3]. All available techniques focus on the server maintenance or to improve the middle layer. Some techniques are used to reduce the number of transactions but the problem of server maintenance still persists. Now we introduce the new technique that uses the JavaScript model to store the information at client side and then send all the transactions made by the user send to the server in a single request. This system helps to develop the forms at run time, rich JavaScript capable to draw the form without compile or upload the page. All the existing technique makes the server scalable but in our purposed system we focus to fetch all the data from database in JavaScript model. This purposed techniques valid for single transaction system and partially offline mode.

2. Related Work

Client server model describes the communication between two instances of computers where one is termed as client who is on service requesting end and other is termed as server which is at service fulfilling end [1]. Client server architecture is all about sharing resources more specifically it's all about sharing server's resources by clients. Normally communication between two instances happens through a certain protocol. That protocol can be termed as Network. Client and server model can be applied within a single system or it can be over a computer network. Client triggers the communication by requesting server's resources while server fulfills entertains client's requests [5].

Client server architecture approach can also be applied in software engineering. Multi tier architecture is one very good example of this phenomenon. In which multiple layers (Presentation, Business logic, and database) can communicate with each other. 3-tier architecture is highly used multi-tier architecture.

Another example of client server architecture in real life can be online form submission in school, College or university. In these type of transactions programs in computer is acting as client while the computer which is saving the forms is server.

Model-View-Controller is an architectural design in software engineering which can be used to separate the software programs in different logical layers. It separates the application layer and business logic layer from user interface. Adopting this approach can leads to less time input for maintenance. If any amendment is required then instead of going for the whole application change only specific tier can be amended.

Client server technique is good enough to be used for efficient communication but there is some limitation which needs to keep in view while implementing this architecture, as this architecture is based on mutual communication of client and server. All the clients are sending requests to server which in returns shares his resources. Too many requests from too many clients can put an impact on server's performance. For any action at client side server has to respond and bring in the latest data. Good client server model can be implemented if server is built with efficient resources. Server should be able enough to respond to all clients' requests on real time basis.

Scaling dynamic web sites has been gained much attention because of growing financial issues. Most of the existing research on scaling web applications focuses on either content replication [4], [5], [7] or dynamic content caching [8] that need site administrator or manual intervention of user. Furthermore, the dynamic content need page fragments via templates and database triggers [4]. Similarly, some current data replications techniques require specialized application uniformity schemes [4], [5], [8] and force the consumer to handle conflicting results [2].

3. Proposed Technique

In the early era of dynamic website development the request/respond architecture was very famous. The browser application used to send synchronous calls to the server and server used to send response against this request. Later on the Asynchronous post back or Ajax gained popularity due to its hidden request/response nature but still the number of request sent to the server had significance impact on the performance of the server. The purposed methodology can be implemented in single transaction system in which work of one user is not affected by other users. In these types of systems users normally work in partial offline mode management software solution are example of such system where we can implement this technique to enforce minimum request on the server and to boost the performance of the server.

When the user logins into the system all its data is fetched into model layer and this model layers classes are then passed to Dynamic JavaScript Code generation layer that is generated Dynamic JavaScript code for the GUI this JavaScript code is basically build the global shared object of Script model layer and then GUI pages are call required to HTML function of that class that is generate dynamic HTML depending upon contents to be displayed on the page.

When the user leaves the page or he wants to save & reset the state of data the JS class model returns the XML with all the changes user have made in the state of data and then this xml is passed to XML parser which parses this XML and fill the model layer classes which can then be passed to Data access layer for further operations in database.

The proposed design method has the following components as shown in Fig. 1:

***GUI*:** This component fetches data from JS Class model in the form of html and displays the content on the page. Other functions like validation, getting input from the user, is performed here and all the actions taken by the user is recorded in JS class model main object of JS class model is shared among all the pages and also redirection is also be performed on the client side to keep this shared object alive.

***Class Model Using JavaScript*:** This component is replica of ERD that is used to tack all the changes users have made in the state of data it received from database. The whole model returns a global object that contains the user data and this object are shared on all the pages. Every change will be highlighted in this object and when user will leave the system or wants to Save and Reset state of data.

***Business Logic Layer*:** The main purpose of this layer is to monitor the incoming and outgoing traffic between GUI and other components. This layer builds the JS class model after getting dynamic code from Dynamic JavaScript Code Generation Layer and when data has to be saved into database it fetches xml sting from JS class model and passes this string to XML parser for further processing.

***Dynamic JavaScript Code Generation Layer*:** This layer gets the classes from model layer and creates JavaScript code to populate data into JavaScript classes. This JS code is thrown on the page dynamically to generate the contents of the page.

***XML Parser*:** The XML parser gets the XML document as an input from GUI and populates all the data in Server side classes of model layer to be saved in database.

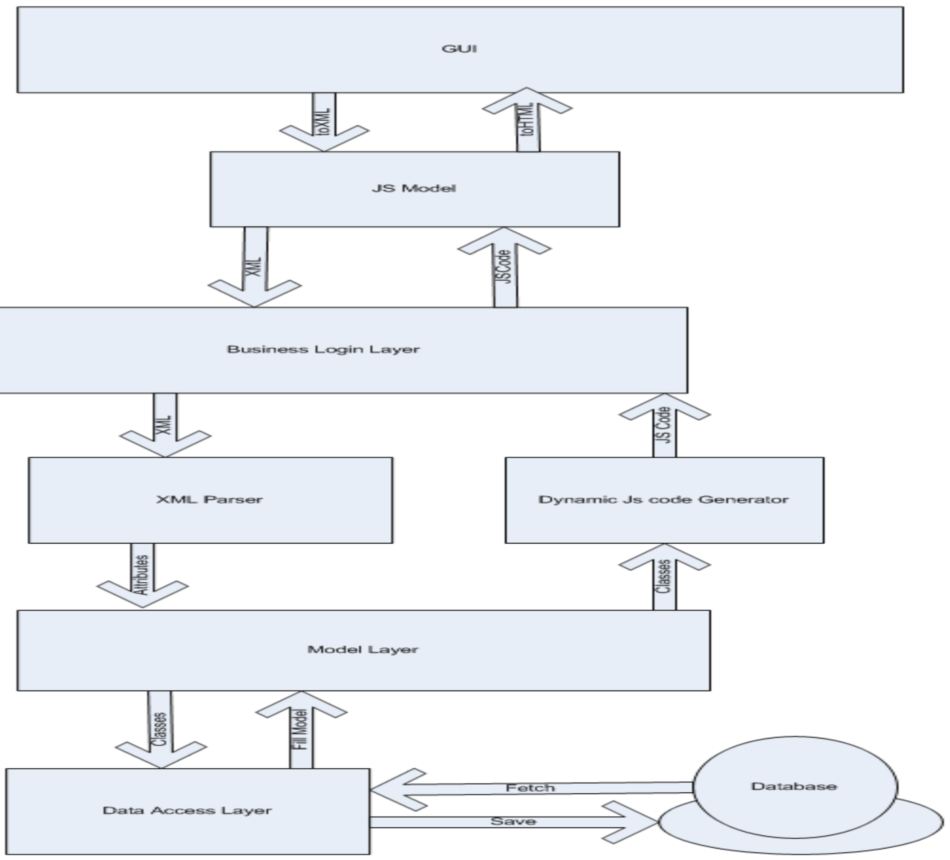

Figure 1: Architecture of the Proposed Technique

***Model layer*:** Model layer models the ERD model on server side classes. These classes have same relationship as database entities and are used to populate to populate user data into memory. If we are using Asp.Net framework then Un-typed dataset are best fit with the methodology.

***Data Access Layer*:** This layer handles all the operations that are relegated to database like fetching data from database. Inserting data into database or deleting data from the database

***Database*:** Database is the essentials parts of dynamic web sites. All data and other supporting information are stored in the database as we store for a normal application. The main point that needs to be more focused while designing database is that all the entities and there relationship should be clearly defined. The same ER model is used to built model layer and Class model using client side scripting. Any change in any entity is needed to be incorporate into model layer and scripting class model.

The proposed technique has very appreciating values of certain parameters such as load balancing, speed, server-interaction and client-intelligent behavior. Using this architecture the number of request sent to the server is less than ordinary request/response architecture so load on the server will be minimized. Everything is directly rendered on client and all the data will be available on client side as well so this will boost the speed of system as well. Request to the server will be sent either on the demand of user or when he will leave the system. Furthermore, all managerial tasks like adding, updating or removing any record will be handled on client side and sate will be saved into database on logout or user's demand so user will be working in partially offline mode.

4. Case Study

To verify the correctness and completeness of proposed technique, we have taken a library management system in which a librarian login into the system and perform various actions like issue new book, update book status and add new student into library etc, the some formal description of class model of case study is listed below:

```
function Grid (id, name, className, rowCount)
{
this.ID = id;
this.Name = name;
this.ClassName = className;
this.Rows = rowCount
this.Columns=new Array ();
this.RowResponse=new Array ();
this.AddRowResponse = function (col)
this.AddColumn = function (col)
  this.toXML = function (mod,SSsequence,label) {}
this.toHTML = function (Mod, Header, FormID, SectionId,
SubSectionID, label) {}
}
function Question (id, text, type, sequence, pageBreak, label)
{
this.ID = id;
this.Sequence = sequence;
this.Text = text;
this.Type = type;
this.Label=label;
this.Options = new Array ();
this.Answers = new Array ();
this.AddAnswer = function AddAnswer (Obj) {}
this.GetAnswerIndex = function (parent) {}
this.GetUniqueAnswerID = function GetUniqueAnswerID
(QuestionID) {}
this.AddOption = function (o) {}
this.GetUniqueOptionID = function GetUniqueOptionID (QuestionID)
{}
this.toXML = function (mod) {}
this.toHTML = function (mode, FormID, SectionId, SubSectionID) {}
}

function SubSection (id, name, className, sequence, type, label)
{
this.ID = id;
this.Name = name;
this.Sequence = sequence;
this.Label = label;
this.Grid = new Grid ('','','',0);
this.Questions = new Array ();

this.AddGrid = function (grid) {}
this.AddQuestion = function AddQuestion (q) {}
this.DeleteQuestion = function DeleteQuestion (id) {}
this.GetQuestionByID = function GetQuestionByID (id) {}
this.SortQuestions=function SortQuestions () {}
this.toXML = function (mod) {}
this.toHTML = function (mode, FormID, SectionID) {}
}

function Section (id,name, className, sequence, pageBreak, label)
{
this.ID = id;
this.Name = name;
this.ClassName = className;
this.Sequence = sequence;
this.Label = label;
this.SubSections = new Array ();
this.AddSubSection = function AddSubSection (SS){}
this.GetUniqueSubSectionID = function GetUniqueSubSectionID
(SectionID) {}
this.DeleteSubSection = function DeleteSubSection (id) {}
this.GetSubSectionByID = function GetSubSectionByID (id) {}
this.SortSubSections = function SortSubSections (){}
this.toXML = function (mod) {}
this.toHTML = function (mode,FormID) {}
       }

function Form(id,title,sequence,istemplate, isComplete)
       {
this.ID = id;
this.Title = title;
this.Sequence = sequence;
this.Sections = new Array ();
this.AddSection = function (s) {}
this.DeleteSection = function DeleteSection (id) {}
this.GetSectionByID = function GetSectionByID (id) {}
this.SortSections = function SortSections () {}
this.toXML = function (mode) {}
this.toHTML = function (mode) {}
}

function Study(id, name, xdate, nResponses, IsTemplate,
ProtocoleNumber, ProjectId, StudyDescription, StudyShortName,
Type, StartDate, EndDate, PlannedEndDate, NextFormID,
NextSectionID, Mode)

{
this.ID = id;
this.Name = name;
this.ExpiryDate = xdate;
this.Responses = nResponses;
this.IsTemplate = IsTemplate;
this.StudyShortName = StudyShortName;
this.Type = Type;
this.StartDate = StartDate;
this.EndDate = EndDate;
this.Forms = new Array ();
this.AddForm = function (f) {}
this.DeleteForm = function DeleteForm (id) {}
this.GetFormIndex = function GetFormIndex (id) {}
this.GetFormByID = function (id) {}
this.toXML = function (mode) {}
this.toHTML = function (mode) {}
}
```

The GUI component can be build dynamically or statically we can use two approaches to build this system either we can create all the forms and can set the value of input/output field from JS Model Layer or we can get whole html from the JS model to show the contents on the page. Client model contains the classes of the entire table that have been created in the database. Of course the relation like inheritance/composition between different entities will also be maintained and all important method are implemented like for book class we implement different data manipulation functions. The Business Logic Layer governs the traffic between GUI and lower layers. it validates the xml that is given to XML parser and contains other business logic. The Dynamic JavaScript Code Generation Layer becomes in action when user will fetch data from database this will create dynamic JS code. This dynamic code will build the JS

Model.

5. Cconclusion aand Future Work

With the enormous growth in web based management solutions the number of transactions increases with the increase in the number of requests. To server so many requests scaling concept was introduced. There are number of scaling techniques available but all these techniques focus on scaling the server by distribute the load on different servers. The purposed study focus on to shift the load on client system and offers a model which can decreases the number of request sent to the server. This Software is best fit with those systems where more than one user does not work on the same data simultaneously. This technique can be further used to create readymade ERPs solutions. The model can be enhanced into two parts one for creation of the website and the other will be for the normal user of system.

Acknowledgment

This research work has been completed with the support of the University of Engineering and Technology, Lahore-Pakistan.

References

[1] Pedro Valderas and Vicente Pelechano, A Survey of Requirements Specification in Model-Driven Development of Web Applications, ACM Transactions on the Web, Vol. 5, No. 2, May 2011.

[2] Acerbis, R., Bongio, A., Brambilla, M., Tisi, M., Ceri, S., and Tosetti, E., Developing eBusiness Solutions with a model driven approach: The case of Acer EMEA. In Proceedings of the 7th International Conference on Web Engineering, pp. 539–544, March 2007.

[3] Atkinson, C. and Kuhne, T., Model-driven development: A meta-modeling foundation. IEEE Softw. Vol. 20, No. 5, pp. 36–41, April 2003.

[4] Brambilla, M., Ceri, S., Fraternali, P., and Manolescu, I., Process modeling in Web applications. ACM Trans. Softw. Engin. Method (ACM TOSEM), pp. 360–409, May 2006.

[5] Cachero, C., Una extensi´on a los m´etodos OO para el modelado y generaci´on autom´atica de interfaces hipermediales. PhD dissertation (in Spanish). Universidad de Alicante. Alicante, Spain, June 2003.

[6] De Troyer, O., Casteleyn, S., and Plessers, P., WSDM: Web semantics design method. Web Engineering: Modelling and Implementing Web Applications. Human-Computer Interaction Book Series, Springer, pp. 303–351, 2008.

[7] Garrigos, I., Gomez, J., Barna, P., and Houben, G. J., A reusable personalization model in web application design. In Proceedings of the 2nd Workshop on Web Information Systems Modelling, (In Conjunction with ICWE'05), August 2005.

[8] Koch, N., Zhang, G., and Escalona, M. J., Model transformations from requirements to Web system design. In Proceedings of the ICWE'06, May 2006.

2/20/2012