OPTIMIZING AND ENHANCING PERFORMANCE OF MVC BASED ON DATA MINING

Miss. Kamble Dhanashri R.¹, Miss. Mirkane Sayli R.², Miss. Kolekar Pritam G.³, Miss. Padalkar Vaishali T.⁴, Prof Ekatpure J.N.⁵
Department of Computer Engineering, S.P.V.P.S.B.P.C.O.E

ABSTRACT:
Electronic web sites have been rendering useful services to organizations ever since the inception of WWW (World Wide Web). Internet has changed the way organizations plan and achieve business goals. Businesses in every conceivable field started relying heavily on web applications to promote sales and services and reach global audience. The last decade has witnessed increased usage of enterprise web applications that bestow on immense benefits to both consumers and service providers. Banking, insurance, healthcare and other domains are exploiting the technologies associated with electronic super high way in order to have competitive edge in the global economy. Electronic commerce and net banking are widely used web applications in the real world. Quality and performance of such web applications play a pivotal role in their sustenance and business continuity. Users of web applications are too good to lose concentration when the applications exhibit mediocre performance. They expect rich user experience in terms of quality and performance. There are some desirable attributes such as response time, throughput, fault tolerance, scalability and availability. In these response times, throughput is performance attributes and fault tolerance, scalability and availability are quality attributes. These attributes can improve quality and performance of web applications.

A web application with robustness, quality and performance can via attract new users and retain existing ones. Many attempts were made as found in the literature to provide various architectural and design patterns to improve quality and performance of web applications. For instance, architectural patterns such as MVC (Model View Controller), JEMSF (Job Evaluation Model Specific Framework), OOHDM (Object - Oriented Hypermedia Design Model), PAC (Presentation Abstraction Control), SMT (Simultaneous Multithreading), PCMEF (Presentation Control Mediator Entity Foundation), ISPWAD (Internet Security Provider Web based Application Design) etc. The common thread identified in these architectural patterns is that each framework is focused on certain aspects of design of web applications. From the analysis of these architectures it is understood that there was no comprehensive architectural pattern with underlying design patterns for improving response time, throughput, fault tolerance, scalability and availability of web applications.

We will try to enhance and optimize the performance of MVC architecture based on two parameters namely response time and throughput. The response time and throughput is improved based on the proposed database search algorithm using B+ tree. If the database search engine is idle, the database search engine will look forward to discover whether the intended data is in level 1, otherwise it will search for level 2 data. The level 2 data will be included as level 1 data inside the database or vice versa, for insertion and update operation. However, whether the data is level 1 or level 2 data will be depended upon user choice. Thus, the overall response time as well as throughput will be optimally increased.
INTRODUCTION

MODEL VIEW CONTROLLER

The important role of MVC is to contrive dynamic software system. MVC is very useful in simplifying the development of bilateral web application. Using the isolated pattern it becomes easy to modify the visual aspect of the application and logic without affecting the other.

Model - Performance and data of the application domain is handled by Model. It ripostes to requests for information which usually comes from the view and ripostes to instructions to change state (usually from the controller). The model component of this architecture represents the data. It basically contains the application data, logic definition, function specification, business rule involvement. This layer not only defines different web data forms but also provide the access objects and the mechanisms. Another consideration is also given to the application or the environment based data analysis so that the service interaction will be done effectively.

View - A view is a (visual) representation of its model. This user interface is described using HTML along with graphical components. This component specification is independent to the browser environment as well as to the application. It generally highlights confirmed attributes of the model and quash others attribute. Thus it acts like a presentation filter. A view is joined to model class and retrieve the data from the model for the presentation. Model can be updated by the view through send appropriate messages.

Controller - A controller is the link to communicate between a user and the system. It retrieves the input from the user then after it gives command to model for processing on it. The controller obtains output, then translates the output into the appropriate messages and conveys these messages through view.

User experience that a web application provides is an integral part of the application’s performance. If customers of a commercial web site experience, slow responsiveness, it is unlikely that they prefer the web site irrespective of other excellent services it offers. Therefore in the best interest of the provider of web application, consistent and rich user experience always is an inevitable requirement. There might be many reasons for inconsistent responsiveness of a web site such as high load, component failure, and so on. However, from user perspective, these are not important as he looks for good service.

The modern era web applications have become such a prevalent phenomenon. In past days Web applications region are broadly not known and it is extremely not understandable fixed appliance due to its complex nature and it is highly technical. Nowadays with the help of web application we can express, manage, depot and transmit the delicate consumer data for
instant and frequent use. Some Web applications take a few seconds and other take huge amount of time to display the results. The work has been done to enhance the performance of Web applications.

Extreme description of the latest developments in the Web is provided by World Wide Web to empowering readers to retrain up-to-date with this dynamic changing technology. New frameworks and platforms for processing have been advanced to vault the web development in past few years. Nowadays Web applications have an important role in every faster that glare the performance concerns. Performance optimization is the most important for faster growing of a web application. Most of the web applications are developed in the client-server architecture. Client - server model is defined as the relationship between two computer programs that communicates with each other.

This needs some sort of consistent and proven solution. Design patterns are such blueprints or proven solutions or industry best practices which can be used to improve the performance of web applications in terms of responsiveness or access time and throughput. Access time [41] is the delay or latency between the time when request is made and the time at which response is rendered. Throughput is used to measure the workload of a web application. In fact, it is best referred to as quantification of requests or responses in relation to time. In other words, it is the number of transactions per second.

Figure 1 presents the MVC pattern which has plethora of advantages including maintainability, availability, and scalability and so on.

As can be seen in fig. 1, every request goes to a controller which is always Servlet that can handle request. However, servlet does contain business logic (BL) and presentation logic (PL). The BL is moved to the model while the presentation logic is moved to view components in order to make the architecture maintainable. Controller invokes BL methods on the model. The Model layer interacts with the database and gives response back to controller. Then the controller invokes suitable view to render response. Due to the clear separation of layers, this architecture realizes many advantages such as maintainability, availability, reduction of development time and cost. However, the MVC gives freedom to use any design patterns in View, Model and Controller layers in order to improve the performance of web applications. This fact has motivated us to propose a new architectural pattern based on MVC. The proposed architectural pattern is elaborated in section 3. Our
contributions in this paper help in designing web applications that exhibit improved performance in terms of access time and throughput. The proposed architectural pattern also supports extension in the future to focus on other performance or reliability parameters like scalability, fault tolerance, availability and maintainability. Our significant contributions are as given below.

1. We proposed an architectural pattern based on MVC with provision for the appropriate design patterns in all the layers for improving access time and throughput. The proposed architectural pattern is extensible and thus we intend to add more design patterns to architecture in future.

2. We evaluated our architectural pattern through case study web application that exhibits the subtle difference in performance between a web application that uses our architecture and the web application that does not use it.

3. Throughput [31] based metric is used to compute average number of responses rendered for given unit time [26]. Latency (response time) is a measure that tells how long user waits to get response to a query. The latency is further divided into two elements known as fetch latency and render time. Fetch latency is the time to load web page into browser while the render time is the time required to receive elements references by the loaded web page [10].

4. The response time and throughput is improved based on the proposed database search algorithm using B+ tree.

In this paper, we will try to enhance and optimize the performance of MVC architecture based on two parameters namely response time and throughput. The response time and throughput is improved based on the proposed database search algorithm using B+ tree. If the database search engine is idle, the database search engine will look forward to discover whether the intended data is in level 1, otherwise it will search for level 2 data. The level 2 data will be included as level 1 data inside the database or vice versa, for insertion and update operation. However, whether the data is level 1 or level 2 data will be depended upon user choice. Thus, the overall response time as well as throughput will be optimally increased.

**PROPOSED SYSTEM**

**Basic Idea of Proposed Method**

The Model-View-Controller (MVC) architecture has its roots in Smalltalk (Sun, 2002). The MVC triad of classes is used to build user interfaces in Smalltalk-80 (Gamma el al., 1995). Three objects make up MVC: Model, View and Controller. The Model is the application object, the View is its screen presentation and the Controller defines how the user interface handles user input. Before MVC was introduced, designs of user interfaces often mixed these three objects together. By separating them you increase flexibility and reuse.

MVC can be mapped to several design patterns and Gamma el al. (1995) describes this in more detail. Each object in MVC is specialized for its task (Burbeck, 1992). The View manages the output of the application. Several Views can be attached to a Model and new
Views can be created without rewriting the Model (Gamma et al., 1995). The Controller handles input (typically keyboard and mouse), telling the Model or the View to change as appropriate (Burbeck, 1992).

Finally, the Model manages the behavior and data of the application domain, answers requests (usually from the View) about its state and listens to instructions (usually from the Controller) to change its state. Both the Controller and the View depend on the Model, but the Model depends on neither the Controller nor the View (Microsoft, 2004). This separation allows the Model to be built and tested independent of the Controller and View. Figure 4.1 shows the structural relationship between the three objects.

This concept is straightforward to map into the domain of enterprise applications (Sun, 2002). The Model then represents enterprise data and the business rules that control access to and updates of this data. The Model aims to be a good software approximation of a real-world process. The View displays the content of the Model (Sun, 2002). It accesses enterprise data through the Model and specifies how data should be displayed. The View is responsible for keeping itself updated when the Model changes. This is done by either the push or the pull model. In the push model, the View registers itself with the Model for change notifications and in the pull model the View is responsible for calling the Model when it needs to retrieve the most current data. The push model is also known as the active model (Microsoft, 2004) and the pull model as the passive model (Burbeck, 1992; Microsoft, 2004). Interactions with the View are translated by the Controller into actions to be performed by the Model (Sun, 2002). Stand-alone GUI clients would have button clicks or menu selections as user interactions.

A Web application has GET and POST HTTP requests. The actions the Model performs are business processes or state changes to it. Depending on the user interaction and the result of the Model action, the Controller will select a new View for response. By using an MVC architecture it is possible to have multiple Views of the same Model (Sun, 2002). New types of clients can be supported by simply writing a new View and some Controller logic to wire them to the existing Model.

In a Web-based application the user interface logic tends to change more frequently than the business logic (Microsoft, 2004). New user interface screens are added or existing layouts are altered. One of the advantages of a Web-based application is that the user interface can change at any time without the need to have to redistribute the application. If user interface logic is combined with business logic it is very easy to introduce errors in the business logic when doing even minimal user interface changes. Often an application displays the same data
in different ways, depending on audience (Microsoft, 2004). The user interface is also more device-dependent than the business logic.

A clean separation of these two parts will make it easier to migrate or extend the application support for new clients, like PDAs and cell phones. Web applications use the passive model of MVC (Microsoft, 2004). The browser (View) displays the presentation and responds to user input, but it does not detect changes in the data (Model) on the server. The user explicitly has to request a refresh. The server-side components handling the HTTP request are the Controller.

The New Component-Based Design

The new architectural design that was done by the author focused on a clear separation by using the MVC paradigm. No better suited architecture has been found. The application is split into a number of components, each of which can be replaced or reused by another application. In this design, the Model is implemented by using Oracle database and all business logic is put there. This means that data integrity is guaranteed by the database component by using various constraints. All retrieval or change of data is done by calling stored procedures (functions). As the Model is dependent on neither the Controller nor the View, it can be developed and tested independent of these.

Basic Idea of Proposed Method:

In this paper we are proposing one of the methods of proposed database clustering technique for improving response time and throughput based on MVC pattern. We identified the need for a simple solution for model layer which can reduce the database access time, query cost and improve throughput and response time.

![Figure 2: type 2 architecture MVC](image)

Framework Description:
SQL Command Scanner:

As described above the job of SQL scanner is to identify the type of SQL command it is (insert, update, delete or read operation). For insert it will directly hit on Primary DB and for other operations it will hit to Secondary DB.

4.2.2 Insert Operation:

Insert Operation directly access Primary Database.

4.2.3 Update, Delete, read Operation:

Update, Delete or Read Operation, it will search Primary DB first and if it is not found there, it will certainly be in Secondary DB.

Configuration:

Configuration will have interface with application layer, where the application developer/user will have options to configure how database file will be cluster, the age classification of data etc. This system configuration will also be used for Database connectivity, clustering, data synchronization with others.

The configuration includes:

1. Database Server Name or IP Address
2. Database Platform: MS SQL SERVER, MySQL, ORACLE, Others
3. Primary & Secondary Database Name
4. Database User
5. Database Password
6. Database Port (Optional)
7. Application Deployment Date & Time
8. Time Frame in Day
9. Clustering Mechanism Configuration

Database Manager:

Database manager will keep records of existed data on the database files. Any data exceeds the time frame given in the global configuration will be shifted from one database file to another (i.e. Primary DB to Secondary DB or vice versa). Any data or record which was
access from the Secondary will be converted as Level 1 data, because according to our classification mechanism any data is accessed recently will be treated as Level 1 Data.

**Primary DB Interface Layer & Secondary DB Interface Layer:**

This layer communicates between database and proposed system

**Algorithm:**

Data Manager (DES: Database Engine Status, DSPDB: Data Set of Primary DB, DSSDB: Data Set of Secondary DB, TP: Time Period)

IF

DES is idle THEN

BEGIN

TL1D← Get Level 1 Data from DSSDB according to age of data & data access frequency

TL2D← Get Level 2 Data from DSPDB according to age of data & data access frequency

Move TL1D to DSPDB

Move TL2D to DSSDB

END

ELSE

Wait until TP;

**Caching**

Cache is a portion of local memory which holds data objects which are frequently accessed from database. Web application performance and scalability can be improved through caching. Caching API helps increase performance in orders of magnitude in terms of response time and throughput. Caching reduces round trips to database by reusing the data present in local memory.

**Architectural Design**
Architectural design

User Browser:
First user sends request using view part. Here view pages are designed using jsp pages.

The Action Servlet:
It is the Controller component that handles the client request and determines which Action class will process each received request. As in MVC2 architecture only one controller component should govern the flow of application - i mean when request is generated ,it should be served by action class and based on the outcome next view is selected (internally Request Dispatcher forward() method is used to choose next view). All business logic is executed by Action class, so Action Servlet does not have such overhead that we need multiple instances.

Model class:
Model is where the application’s data objects are stored. The model doesn’t know anything about views and controllers. When a model changes, typically it will notify its observers that a change has occurred.

Action: actions are nothing but no. of user requests. he Model layer represents the part of your application that implements the business logic. It is responsible for retrieving data and converting it into meaningful concepts for your application. This includes processing, validating, associating or other tasks related to handling data.

The View layer:
The View renders a presentation of modeled data. Being separated from the Model objects, it is responsible for using the information it has available to produce any presentational interface your application might need.

For example, as the Model layer returns a set of data, the view would use it to render a HTML page containing it, or a XML formatted result for others to consume.
The View layer is not only limited to HTML or text representation of the data. It can be used to deliver a wide variety of formats depending on your needs, such as videos, music, documents and any other format you can think of.

**Struts.config.xml:**

It is nothing but configuration file used in system

**Database manager:**

Will keep records of existed data on the database files. Any data exceeds the time frame given in the global configuration will be shifted from one database file to another (i.e. Primary DB to Secondary DB or vice versa). Any data or record which was access from the Secondary will be converted as Level 1 data, because according to our classification mechanism any data is accessed recently will be treated as Level 1 Data.

**RESULT**

<table>
<thead>
<tr>
<th>id</th>
<th>firstname</th>
<th>surname</th>
<th>email</th>
<th>resent</th>
<th>password</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>n</td>
<td>m</td>
<td>k</td>
<td>k</td>
</tr>
<tr>
<td>2</td>
<td>global</td>
<td>global</td>
<td><a href="mailto:global@gmail.com">global@gmail.com</a></td>
<td>global</td>
<td>global</td>
</tr>
<tr>
<td>3</td>
<td>saner</td>
<td>sa</td>
<td><a href="mailto:saner@gmail.com">saner@gmail.com</a></td>
<td>saner</td>
<td>saner</td>
</tr>
<tr>
<td>4</td>
<td>Raju</td>
<td>kajal</td>
<td><a href="mailto:raj@gmail.com">raj@gmail.com</a></td>
<td>global</td>
<td>global</td>
</tr>
<tr>
<td>5</td>
<td>j</td>
<td>h</td>
<td>p</td>
<td>n</td>
<td>n</td>
</tr>
</tbody>
</table>
Thanks for Registering with us:

To login with new UserID and Password: Click here

Submit project details: Click here Project Details

Registration Form

First Name: [Input field] (max 30 characters a-z and A-Z)
Middle Name: [Input field] (max 30 characters a-z and A-Z)
Last Name: [Input field] (max 30 characters a-z and A-Z)
Email: [Input field]
UserID: [Input field]
Password: [Input field]

Submit | Reset
Project Record

Enter Project Name (max 30 characters a-z and A-Z)

Message from webpage

Enter Project Name!

OK

Submit Reset

**Search Student Details**

Student information
Thu Apr 27 16:32:03 ET 2017

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Front End Language</th>
<th>Back End Language</th>
<th>Name of Database</th>
<th>No of Modules</th>
<th>Application Type</th>
<th>Project Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVC</td>
<td>jsp</td>
<td>servlet</td>
<td>MYSQL5</td>
<td>10</td>
<td>Web Application</td>
<td>Improve the performance of mvc based application with database</td>
</tr>
</tbody>
</table>
Project Record

Enter Project Name: MVC (max 30 characters a-z and A-Z)

Frontend Language: jsp (max 30 characters a-z and A-Z)

Backend Language: servlet (max 30 characters a-z and A-Z)

Database name with version: MYSQL 5

No. Of Modules: 10

Application Type: Web Application

Project Description: Application with database

Submit Reset
CONCLUSION

CONCLUSION AND FUTURE WORK

Conclusions In this research, we proposed a novel architectural pattern for web application development. The architecture includes many design patterns that help improve access time and throughput of web applications. The design patterns are pertaining to database access, connection pooling, caching and so on. The whole architecture is based on the MVC pattern. MVC gives freedom to improve layers such as Model, View and Controller further without violating their assumed roles. We were motivated by this fact and proposed a novel architectural framework XWADF for improving performance of web applications. We have tested the architecture using state-of-the-art web application development with and without the usage of our architecture.

Idea of splitting an entire database is a novel idea. Append operation is much more common than query operation whereas update operation happens almost on the odd occasion. In cases where data are stored and after certain time those data are rarely or not accessed at all for long time, our frame will definitely reduce overhead on database engine by reducing volume of data in the primary database. Hence, it will increase overall response time and throughput of the database. The above experimental results show a certain improvement in performance of our system. In future, we will try to embed such an algorithm that will be feasible for process industry where data change happens very frequently.
FUTURE WORK

The future directions for further improvement of the architecture includes the extension of the proposed architecture gradually in order to include more design patterns and services that can leverage web applications in terms of quality and performance of web applications. One of the additional design patterns that will take in proposed architecture is for maintainability. To improve maintainability we can consider different design patterns for maintainability and enhance the XWADF architectural pattern.

In future, we will try to embed such an algorithm that will be feasible for process industry where data change happens very frequently.

We can improve functionality by considering design patterns related to this functionality. All these design patterns are taken in XWADF to improve functionality. Likewise we can take different design patterns related to efficiency in XWADF and improve the efficiency of web applications. In this way we can enhance the XWADF for different design patterns.

References


