XWADF: Architectural Pattern for Improving Performance of Web Applications

Md. Umar Khan[1], Dr. T.V. Rao[2]
[1]Assoc. Professor, Prakasam Engineering College, Kandukur, AP, India

[2]Professor& HOD in CSED, PVP Siddartha Institute of Technology, Kanuru, Vijayawada, AP, India

Abstract
Ever since the advent of World Wide Web (WWW) web sites and their usage has become part of day-to-day life. Enterprises reach global audience through web applications. People of all walks of life need to use web applications in one way or other. Performance of web applications plays a key role in attracting new users and retaining existing ones. In this paper, we investigate the design patterns that can improve performance of web applications. We propose a new architectural pattern comprising design patterns for highly scalable and interactive web application development. Our architecture is known as extensible Web Application Development Framework (XWADF). Our design does not reinvent the wheel, but explores possibilities to leverage the performance of web applications through the appropriate use of various design patterns within confines of MVC (Model View Controller). Particularly we throw light into the performance of web applications by improving response time and throughput. We use corresponding metrics to evaluate the efficiency of the proposed architectural pattern. The empirical results revealed that our approach to design web applications outperforms existing approaches.
Keywords: Web applications, design patterns, architectural pattern, response time, and throughput

1. Introduction

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.

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.

Technological innovations over WWW such as Web 2.0 [35] facilitate designing web applications with rich user experience. For instance, asynchronous communication through AJAX (Asynchronous JavaScript and XML) will let the servers push content to web browsers without full page refreshes. Thus with AJAX [30] rich internet applications can be built. Nevertheless, we believe that consistent performance of the web application can be leveraged by the appropriate use of design patterns. However, there is a fact to be kept in mind that resources associated with web server can influence the performance of a web application. Assuming optimal resource availability of web server, design patterns can improve performance of web applications in terms of access time and throughput.

This is the motivation behind this paper which focuses on proposing an architectural pattern [42] for improving performance of web applications. Our future research continues with other parameters such as scalability [36], fault tolerance [37], availability [38] and maintainability [39]. Our work in this paper also assumes the usage of three-tier architecture, Java enterprise technologies like Servlets and JSP [33] (Java Server Pages) as web tier under optimal resources availability of web server. The proposed architectural pattern is based on Model View Controller (MVC) [19] architecture which is widely used in the industry for enterprise web application development.

The proposed architectural pattern focuses on the appropriate usage of design patterns [40] in different layers of MVC (Model View Controller). We identified various services and design patterns that can improve the performance of web application in terms of latency and throughput. However, architecture is extensible to consider other performance and quality attributes such as scalability, fault tolerance, availability and maintainability in future. Before presenting our architectural pattern, we felt the description of basic MVC architecture is appropriate here. Figure 1 presents the MVC pattern which has plethora of advantages including maintainability, availability, scalability and so on.

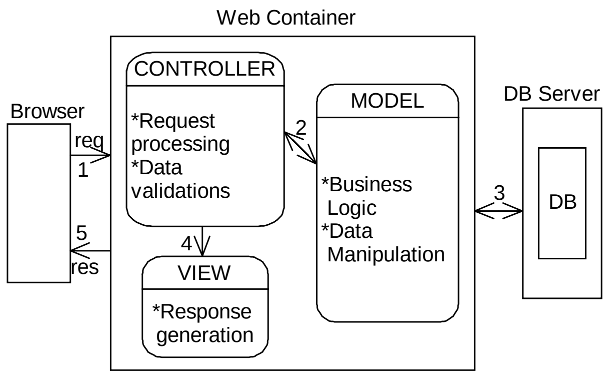

Fig. 1 –Illustrates MVC pattern

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 the 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 tell 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. We also provide a roadmap algorithm known as "Refactoring Algorithm" which helps developers to upgrade their enterprise web applications in conformity with the proposed architectural pattern for performance gains in terms of throughput and response time.

The remainder of this paper is organized as follows. Section 2 reviews literature pertaining to architectural patterns and design patterns that improve access time and throughput of web applications. Section 3 presents the proposed architectural pattern. Section 4describes the case study considers for proof of the concept. The section 5 evaluates the proposed architectural pattern with respect to web application performance improvement while section 6 concludes the paper.

2. Related Works

Many researchers attempted to propose frameworks or patterns for web application design that improves performance. LIU YONG-JUN and LI KE-XI [1] proposed a new web application framework by name JEMSF. This framework is based on the MVC pattern which is widely used in web application development in all platforms. Its main focus was on the usage of the data transfer object, and database access object patterns besides using a linked pool for avoiding scarcity of database connections. They also proposed the usage of a configuration file that is used by the controller at runtime. For database access also they proposed another configuration file for making the web application to have flexibility to switch between RDBMS. The JEMSF has three layers. The first layer makes certain events and gets responses. The second layer is the controller that performs XML [20] interpretation, request processing and exception handling. The third layer is responsible to interact with the relational database. It has provision for pooling objects to reuse and SQL processor component that will interact with database through database objects obtained from the pool.

Schwabe et al. [3] proposed a model based approach named Object - Oriented Hypermedia Design Method (OOHDM) for building large scale hypermedia applications. This model comprises of four kinds of things namely conceptual design, navigational design, interface design at abstract level and implementation. They used object oriented modeling principles in designing their architecture. The conceptual model encapsulated classes. The navigational classes take care of dynamic navigations in web application. Interface objects are focused on abstract interface design activity while implementation integrates all these into a working solution.

Tune et al. [4] proposed a simultaneous multithreading (SMT) processor for increasing throughput of applications. Thus, the SMT can make use of maximum utilization of resources. Broadwell [5] did experiments on response time [32] as performance metric to evaluate the performance of internet services. David Parsons [6] presented evolution of architectural patterns for developing enterprise web applications. They include client side buffering, delta-management architecture, and auto-completion architecture. Patil et al. [7] opined that modern web application developers should consider achieving high latency of web applications to maximize user experience. They proposed the proxy cache concept for enterprise application development. As explored in [8] caching is best used when the same payload is carried by multiple queries. Event the commercial user agents like Firefox, Chrome, etc. also have cache-friendly features [9]. Feng and Yang [10] presented an improved connection pool model that can be used in modern web applications for improving latency and throughput. They proved that improving connection pool performance can lead to web application performance. A web application design method was proposed by Kwon and Bang [11] for performance improvement. In their design BL is implemented in JSP search operations are implemented in Servlet. A forward servlet was also used to avoid maintenance problems. Mamawala [12] studied on performance of web applications and concluded that as the time changes performance is deteriorated while the number of users and features are constantly increased. He further stated that balancing performance objectives with the usability, functional and quality of service aspects have to be done from time to time. Thung et al. [13] explored the usage of design patterns for performance and quality improvement of web applications. Both navigational and architectural patterns were studied by them. They include MVC, PAC (Presentation Abstraction Control),pagination, set– based navigation, news, the navigation observer and so on.

According to Li and Lu [14] performance of a web database has its influence in the performance of a web application. They considered response time to measure the performance of web applications. In [15] Dong et al. identified design patterns based on the modeling such as UML notations. They achieved it using a tool which results in improving the performance of web applications. Maciaszek and Liong [16] proposed an architectural pattern for web application development. They followed a layered architecture and named it as “PCMEF”. It has four layers, namely presentation, control, domain and foundation. The presentation layer is to provide user interface. The control layer is responsible for request processing. The domain layer has two packages such as an entity and mediator for encapsulating business objects and to mediate between control, the entry package and foundation layer respectively. The foundation layer provides communication with data sources. Madeyski and Stochmiałek [17] proposed a new architectural known as the extensible Web Architecture (XWA) framework for web application development. This framework is based on both MVC and PCMEF. It also has four layers namely presentation, control, domain and foundation. However, these layers are organized into MVC so as to mix the MVC and PCMEF architectures.

3. Proposed Architectural Pattern

This section throws light into the proposed architectural pattern for web application development for improving response time and throughput. The pattern is based on MVC described in section I. Our architectural pattern incorporates various design patterns into M, V, and C layers. We identified the need for a simple solution for the model layer which can reduce the communication cost and improve throughput and response time. We also explored connection pooling configuration, caching, the decorator design pattern [18], Data Transfer Object (DTO), Database Access Objects (DAO) and other patterns. Figure 2 shows our architectural pattern.

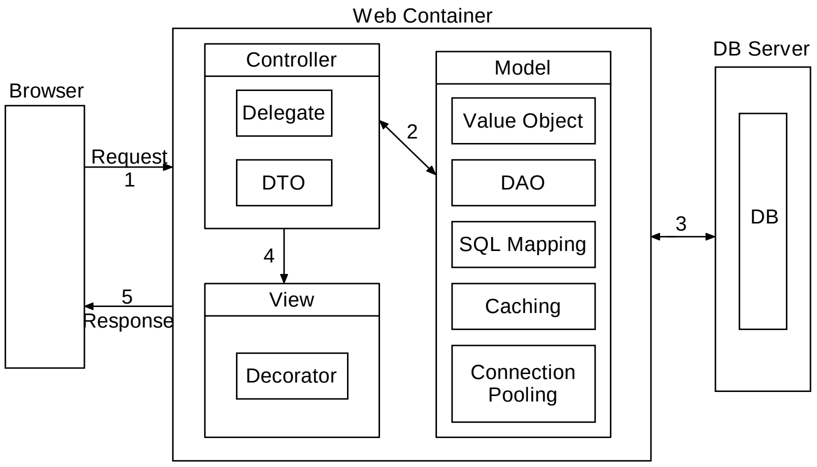

Fig. 2 –Proposed architectural pattern XWADF

In all layers of XWADF design patterns are proposed to improve throughput and response time of web applications. These patterns have the plethora of advantages. However, in this paper we focus on their ability to improve response time and throughput. In the Model layer design patterns are used for returning values, database interaction, SQL mapping, caching and connection pooling.

3.1 Value Object or Active Record or DTO

This pattern is best used when the underlying data storage is a relational database. The properties of this object correspond to columns in the relational table. Thus it is suitable to hold a return value from the database. When multiple records are to be returned multiple instances of this object can be added to a collection and returned to the controller. The SQL Mapper program can map the results of queries to Value Object or collection automatically. This will help in avoiding extra boiler plate coding in the application development [21].

3.2 Database Access Objects (DAO)

This design pattern separates low level data access operations from high level business services. Thus it can Be reused in applications as it is separated from other layers. It is dedicated for database interaction only. It is made up of three participants namely DAO interface, DAO implementation and VO or DTO or POJO [43]. The POJO is meant for either data transfer from other layers to DAO or to return values back to higher layers [24].

3.3 SQL Mapping

We introduce a special design pattern that maps results of SELECT queries to required data objects like POJOs, collections of various kinds or simple primitive type. A design pattern that ensures this kind of mapping can help developers to reduce development effort and time by avoiding reinventing the wheel every time. By reducing the lot of boilerplate code in web applications, this design pattern can improve the speed of web applications [27].

3.4 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. Figure 3 shows communication diagram to illustrate the caching mechanism [22].

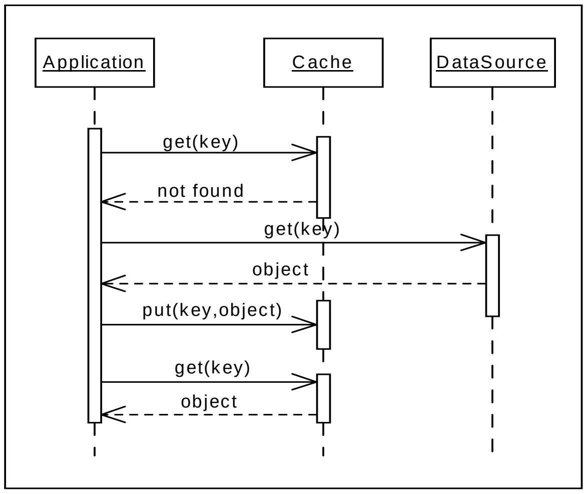

Fig. 3 Communication diagram illustrating caching

Application hits the database only when the object required is not in the cache memory. Since round trip to database is time consuming and costly, this solution can improve the performance of web applications. The performance of cache is measured using hit/miss ratio which is computed as number of cache hits divided by number of cache misses. A high hit/miss ratio reflects high performance of cache.

3.5 Connection Pooling

Establishing connection to the database is costly and time consuming as it involves in a series of steps including protocol handshaking. Closing and opening database connections are not desirable. Moreover, database vendors provide less number of connections per schema by default. These connections are exhausted as web applications are used by the number of users concurrently.

To overcome this problem connection pooling is required. Connection Pooling is a phenomenon that maintains a set of pre-established connections to a database in a pool. These connections are never closed. They are available round the clock. When the application needs a connection, the connection pool manager gives a connection.

Once the request processing is completed, the connection goes back to the pool and ready to serve the next request. This way the performance of web applications in terms of response time and throughput is increased in large numbers of magnitude. Connection pooling usage can be compared with normal database connectivity as illustrated in figure 4 [25].

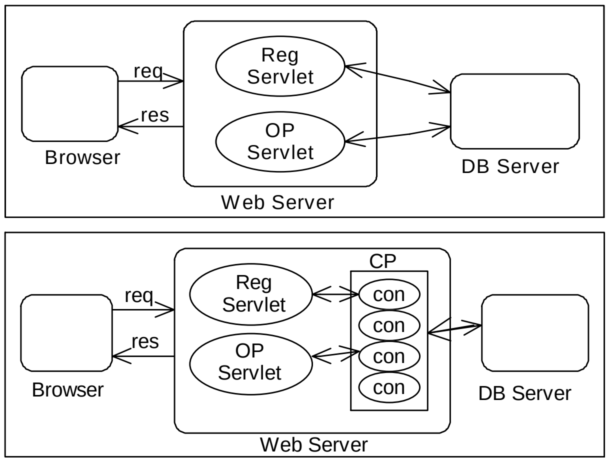

Fig. 4 Illustrates DB connectivity with and without pooling

In case of direct connectivity, user wait time is increased, connections scarcity arises and overall application performance goes down. With connection pooling these drawbacks are overcome as it makes the application more responsive and scale well for concurrent request processing. Connection pooling is recommended for high performance of enterprise web applications [44].

3.6 Delegation Design Pattern

Delegation design pattern helps an object to delegate its tasks to other objects instead of doing itself. This will help in achieving inversion of responsibility or to pass the buck for making the design efficient. It reduces the complexity of the application as it follows the division of labor approach to divide the work into various helper classes [23].

3.7 Decorator Design Pattern

This design pattern [28] helps in improving the abilities of the runtime object without making any changes in source code or modifying other instances. Decorators are very useful to add new responsibilities and features to runtime objects. When only some of the objects need new features this design pattern is preferred. Thus it can avoid having large number of sub classes in order to add new features to the object. Decorator provides a lot of flexibility in adding features to existing objects [18].

4. Converting Existing WEB Applications to XWADF: A RoadMap

Existing web applications that do not perform well can be refactored to be XWADF compliant in order to leverage their performance. We designed a refactoring algorithm that can help convert existing web applications into XWADF compliant applications. The algorithm guides developers to refactor successfully. This algorithm can be a basis for developing an automatic conversion tool in the future. Figure 5 shows the proposed refactoring algorithm [29].

```
Algorithm :Refactoring Algorithm
Inputs :Existing Web Application
Outputs : XWADF Web Application
Assumptions :Java Web Application with Servlets and
             JSPs as Web Resources
Process
  1. START
  2. If Web App is in MVC Pattern Then
     a. Configure connection pool in
        web server/application server
     b. Implement, a design pattern that
        gets connection from pool
     c. Define POJO for every relational table
     d. Move the DB interaction logic to DAOs
     e. Implement SQL Mapper design pattern
        and use it in DAOs
     f. Implement Design Pattern for Caching
     g. Apply caching in the Model layer
     h. Use Delegation Pattern and DTO in controller
     i. Identify responses to the decorator in the view
     j. Use decorator pattern to decorate responses
  3. If Web App is not in MVC Pattern Then
     a. Move the presentation logic to view layer
     b. Move business logic layer to model layer
     c. Go back to step 1 to refactor as specified.
  4. STOP
```

Fig. 5 Refactoring algorithm (Roadmap to XWADF)

The refactoring algorithm guides developers to refactor their existing web applications to be compliant with our architectural framework XWADF. We have taken two case studies to demonstrate the effectiveness of the proposed architectural framework with suggested design patterns.

5. Case Study and Experiments

5.1 Experimental Setup

Experiments are carried out in a PC with 4 GB RAM, Core 2 dual processor running Windows 7 operating system. Tomcat 7.0 is used for the deploying web application. Servlets and JSP are the technologies used in the application.

The performance is tested manually and also with LoadUIWeb 2 which is one of the testing tools available. This tool provides access time and throughput in the presence of the number of users concurrently accessing the web application.

5.2 Case Study Web Applications

We considered the existing web applications, namely "Library Management System" and "Hospital Management System" for our study.

These two applications are refactored to implement proposed architectural pattern. CRUD [45] operations are tested with both case studied with and without design patterns. The dataset of HMS has 100000 entries while that of LMS has 50000 entries. The LoadUIWeb 2 testing tool is used to measure response time and throughput of applications. The experimental results are presented in the next sub section.

5.3 Experimental Results

Experiments are performed with Software tool LoadUIWeb 2 to find latency and throughput of both web applications with and without XWADF. The results are recorded for the single user, 2 users 3 users and so on up to 50 users and tabulated them as shown in table 1, 2, 3, and 4.

Table 1 : Experimental results for Latency of HMS (Hospital Management System)

No. of users	Response Time in seconds	
	Without XWADF	With XWADF
1	74.8	50.33
2	76.9	56.66
5	84.62	60.53
10	95.32	71.32
15	107.23	82.46
20	118.32	94.22
25	130.42	116.43
30	142.36	127.32
35	155.42	138.61
40	172.67	151.32
45	184.32	163.13
50	196.42	174.21

Latency Results for HMS (Hospital Management System) are shown in the Table 1. Latency (response time) is a measure that tells how long user waits to get response to a query.

The response time in seconds for without XWADF and with XWADF is tabulated. The results reveal that there is considerable improvement in response time when the application with XWADF is used for experiments.

Table 2 : Experimental results for Latency of LMS (Library Management System)

No. of users	Response Time	
	Without XWADF	With XWADF
1	20.33	15.45
5	24.2	18.06
10	32.92	27.33
15	44.92	37.78
20	54.11	48.26
25	65.21	59.11
30	78.22	70.12
35	90.15	81.25
40	112.35	92.15
45	125.11	105.41
50	134.51	115.31

Latency Results for LMS (Library Management System) is shown in Table 2. Latency (response time) is a measure that tells how long user waits to get response to a query.

The response time in seconds for without XWADF and with XWADF is tabulated. The results reveal that there is considerable improvement in response time when the application with XWADF is used for experiments.

Table 3 : Experimental results for Throughput of HMS (Hospital Management System)

Time in minutes	Throughput	
	Without XWADF	With XWADF
1	0.01337	0.01987
2	0.01300	0.01765
5	0.01182	0.01652
10	0.01049	0.01402
15	0.00933	0.01212
20	0.00845	0.01061
25	0.00767	0.00859
30	0.00702	0.00785
35	0.00643	0.00721
40	0.00579	0.00661
45	0.00543	0.00613
50	0.00509	0.00574

Throughput results for HMS (Hospital Management System) are shown in Table 3. Throughput is used to compute the average number of responses rendered for given unit time.

Throughput without XWADF and with XWADF is tabulated. The results reveal that there is considerable improvement in throughput when the application with XWADF is used for experiments.

Table 4 : Experimental results for Throughput of LMS (Library Management System)

Time in minutes	Throughput	
	Without XWADF	With XWADF
1	0.04919	0.06472
5	0.04132	0.05537
10	0.03038	0.03659
15	0.02226	0.02647
20	0.01848	0.02072
25	0.01534	0.01692
30	0.01278	0.01426
35	0.01109	0.01231
40	0.00890	0.01085
45	0.00799	0.00949
50	0.00743	0.00867

Throughput results for LMS (Library Management System) are shown in Table 4. Throughput is used to compute average number of responses rendered for given unit time.

The throughput without XWADF and with XWADF is tabulated. The results reveal that there is considerable improvement in throughput when the application with the proposed architectural pattern is used for experiments.

The experimental results are visualized through a series of graphs as presented below.

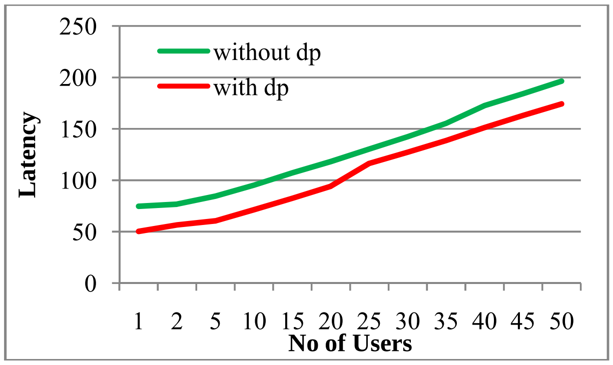

Fig. 6 Latency Graph for HMS (without design patterns (dp) and with design patterns (dp))

As can be seen in Fig. 6 it is evident that the latency is more when experiments are made with HMS without the proposed architectural pattern when compared with the latency of HMS with the architectural pattern.

The rationale behind this is that the architectural pattern includes various design patterns that leverage the performance of the application.

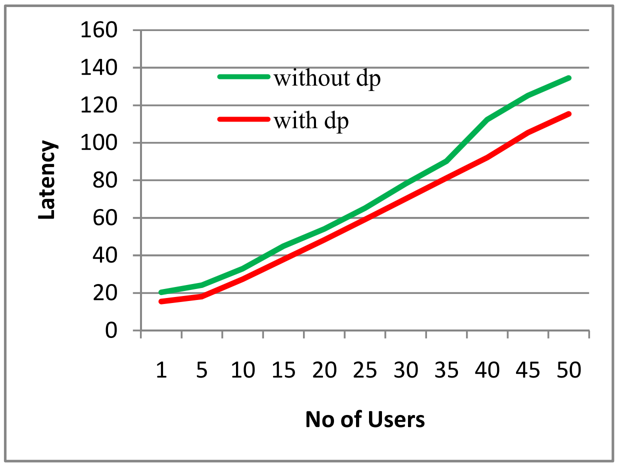

Fig. 7 Latency Graph for LMS (Library Management System)

As can be seen in Fig. 7, it is evident that the latency is more when experiments are made with LMS without the proposed architectural pattern when compared with the latency of LMS with the architectural pattern.

The rationale behind this is that the architectural pattern includes various design patterns that leverage the performance of the application

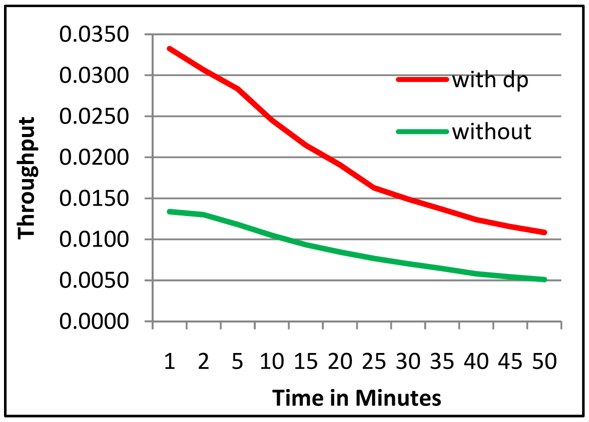

Fig. 8 Throughput Graph for HMS (without design patterns (dp) and with design patterns (dp))

As can be seen in Fig. 8, it is evident that the throughput is less when experiments are made with HMS without the proposed architectural pattern when compared with the latency of HMS with the architectural pattern.

The rationale behind this is that the architectural pattern includes various design patterns that leverage the performance of the application.

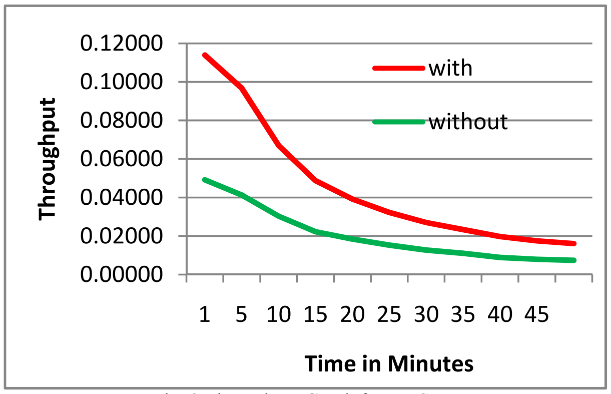

Fig. 9 Throughput Graph for LMS

As can be seen in Fig. 9 it is evident that the throughput is less when experiments are made with LMS without the proposed architectural pattern when compared with the latency of LMS with the architectural pattern.

The rationale behind this is that the architectural pattern includes various design patterns that leverage the performance of the application

6. Conclusions and Future Work

In this paper, we presented 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 which has become de facto standard for enterprise web application development. Besides bestowing many advantages 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. We measured the performance of web application, both manually and also with LoadUIWeb 2 testing tool that simulated the performance of application in the presence of the increased number of sessions and concurrent access. Latency and Throughput are the two measures employed to know the performance of web applications. The experimental results revealed that the proposed architecture improves performance by order of magnitude in terms of latency and throughput. 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 with additional attributes such as scalability, fault tolerance, availability and maintainability.

References

[1]. LIU YONG-JUN1, LI KE-XI, “ DESIGN AND IMPLEMENTATION OF THE NEW WEB APPLICATION FRAMEWORK—JEMSF”, IEEE, 2010.pp 190-193.
[2]. Vijay K Kerji, “Decorator Pattern with XML in Web Application”, IEEE, 2011. Pp304-308.
[3] Daniel Schwabe, Rita de Almeida Pontes and Isabela Moura. (n.d). OOHDM-Web: An Environment for Implementation of Hypermedia Applications in the WWW. *MCT*. 0 (0), p1-14.
[4]. Eric Tune Rakesh Kumar Dean M. Tullsen and Brad Calder, “Balanced Multithreading: Increasing Throughput via a Low Cost Multithreading Hierarchy”. IEEE, 2004.Pp 1-12.
[5]. *Peter M. Broadwell,* “Response Time as a Performability Metric for Online Services”. Computer Science Division, 2004, pp1-54.
[6]. David Parsons, “Evolving Architectural Patterns For Web Applications”. Pp1-7.
[7]. Prof. S B Patil, Mr. SachinChavan, Prof. PreetiPatil and Prof. Sunita R Patil,” HIGH QUALITY DESIGN TO ENHANCE AND IMPROVE PERFORMANCE OF LARGE SCALE WEB APPLICATIONS”.**IJCET, 2012.Pp 198-205.”**
[8] Ngamsuriyaroj, S. ;Rattidham, P. ; Rassameeroj, I. ; Wongbuchasin, P. ; Aramkul,
N. ;Rungmano, S. “Performance Evaluation of Load Balanced Web Proxies”
IEEE, 2011.
[9] S B Patil, SachinChavan, PreetiPatil; “High Quality Design And Methodology
Aspects To Enhance Large Scale Web Services”, International Journal of
Advances in Engineering & Technology, 2012, ISSN : 2231-1963
[10]. Guo-liangFeng, and Lian-he Yang, “A New Method in Improving Database Connection Pool Model”. World Academy of Science, Engineering and Technology 29 2007, pp246-249.
[11] OhSoo Kwon and HyeJa Bang. (2012). Design Approaches of Web Application with Efficient Performance in JAVA. *IEEE*.11 (7), p1-7.
[12] Mustafa Mamawala. (n.d). Web Application Performance Tuning -A systematic approach. *Blue Star*. 0 (0), p1-11.
[13] Phek Lan Thung, Chu Jian Ng, Swee Jing Thung, Shahida Sulaiman. (2010). Improving a Web Application Using Design Patterns: A Case Study. *IEEE*. 0 (0), p1-6.
[14] AKAMAI. (n.d). Web Application Accelerator. *Akamai Technologies*. 0 (0), p1-2.
[15] Martin Fowler and David Rice. (2002). Patterns of Enterprise Application Architecture. *Addison Wesley*. 0 (0), p1-430.
[16] Maciaszek L.A., Liong B.L., Bills S., *Practical Software Engineering*, A Case-Study Approach, Addison-Wesley, 2004.
[17] Lech MADEYSKI and Michał STOCHMIAŁEK, ARCHITECTURAL DESIGN OF MODERN WEB APPLICATIONS, p1-11
[18] Javapapers. (2012). *Decorator Design Pattern.* Available: http://javapapers.com/design-patterns/decorator-pattern/. Last accessed 20th Dec 2013.
[19] Berkeley. (2004). Model-View-Controller: A Design Pattern for Software.*Berkeley.edu*. p1-48.
[20] MSDN. (2004). *Improving XML Performance.* Available: http://msdn.microsoft.com/en-us/library/ff647804.aspx. Last accessed 22 Dec 2013.
[21] Riehle. (2006). Value Object. *Hillside*. p1-9.
[22] Cacheonix. (n.d). *Caching for Java Applications.* Available:

www./articles/Caching_for_Java_Applications.htm. Last accessed 22 Dec 2013.
[23] Princeton. (n.d). *Delegation pattern.* Available: http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Delegation_pattern.html. Last accessed 22 Dec 2013.
[24] Javarevisited. (2013). Data Access Object (DAO) design pattern in Java Read more: http://javarevisited.blogspot.com/2013/01/data-access-object-dao-design-pattern-java-tutorial-example.html#ixzz2rx66gODo.*javarevisited.blogspot*. p1-3.
[25] OODesign. (n.d). *Object Pool.* Available: http://www.oodesign.com/object-pool-pattern.html. Last accessed 22 Dec 2013.
[26] K. Nagaraja, X. Li, B. Zhang, R. Bianchini, R. Martin, and T. Nguyen.Using Fault Injection to Evaluate the Performability of Cluster-Based Services. In *4th USENIX Symposium on InternetTechnologies and Systems (USITS)*, Seattle, WA, March 2003.
[27] Juan Sequeda, Freddy Priyatna and Boris Villaz. (2013). Relational Database to RDF Mapping Patterns. *Ontologydesignpatterns*. p1-12.
[28] Mira Mezini. (2012). Design Patterns Decorator. *Software Technology Froup*. 0 (0), p1-20.
[29] JL OVERBEY. (2013). Immutable Source-Mapped Abstract Syntax Tree: A Design Pattern for Refactoring Engine APIs. *Hillside*. 0 (0), p1-8.
[30] Mrkpinedahau. (2013). Introduction to Javascript and Ajax. *Wikispaces*. 0 (0), p1-31.
[31] Cseweb. (2012). CS423 Computer Communications: More Error Recovery. *Cseweb.ucsd.edu*. 0 (0), p1-12.
[32] James Goodwill. (2012). Pure JSP. *Portal*. 0 (0), p245.
[33] Tutorialspoint. (2012). Servlets. *Tutorialspoint*. 0 (0), p1-180.
[34] John Musser. (2012). Web 2.0 Principles and Best Practices. *Oreilly*. 0 (0), p1-8.
[35] Kanwardeep Singh Ahluwalia. (2007). Scalability Design Patterns.*Hillside*. 0 (o), p1-22.
[36] Luciane Lamour Ferreira and Cecília Mary Fischer Rubira. (2012). Reflective Design Patterns to Implement Fault Tolerance. *Citeseerx*. 0 (0), p1-5.
[37] Kanwardeep Singh Ahluwalia and Atul Jain. (2006). High Availability Design Patterns. *Hillside*. 0 (0), p1-20.
[38] P´eter Heged˝us, D´enes B´an, Rudolf Ferenc and Tibor Gyim´othy. (2012). Myth or Reality? Analyzing the Effect of Design Patterns on Software Maintainability. *Inf.u*. 0 (0), p1-8.
[39] S. Ramachandran. (2001). *Design Patterns for Optimizing the Performance of J2EE Applications.* Available: http://www.oracle.com/technetwork/articles/javaee/j2eepatterns-136748.html. Last accessed 22 Dec 2013.
[40] Bruce Powel Douglass. (2012). Real-Time Design Patterns.*Archive.oredev*. 0 (0), p1-104.
[41] Paris Avgeriou and Uwe Zdun. (2012). Architectural Patterns Revisited – A Pattern Language. *Infosys.tuwien*. 0 (0), p1-39.
[42] Chris Richardson. (2012). Overview of POJO programming. *Richardson*. 0 (0), p1-48.
[43] Cristian Duda, David A. Graf and Donald Kossmann. (2007). Predicate-based Indexing of Enterprise Web Applications. *Cidrdb*. 0 (0), p102-107.
[44] Cesare Pautasso. (2012). Some REST Design Patterns. *Jopera*. 0 (0), p12-44.

Dr. T.Venkateswara Rao received his B.E. Degree in Electronics and Communication Engineering from Andhra University, Visakapatnam, India and his M.E degree in Computer Science from University of Madras India. He received his Ph.D., degree in computer engineering from Wayne State University, Detroit, U.S.A. He is currently working as Professor and HOD in Computer Science and Engineering department at PVP Siddhartha Institute of Technology, Vijayawada A.P. India. Dr. T.V. Rao has published more than 25 papers in various national and international journals/conferences. His main research interests include multiprocessor systems.

Md.Umar Khan received his B.E degree in Civil Engineering from Madras University, Tamil Nadu, and India, his M.Tech degree in Computer Science from Jawaharlal Nehru Technological University, Hyderabad, and A.P. India. He is now pursuing his PhD degree at JNTU Ananthapur University, Andhra Pradesh. His research interests include Web Engineering, especially the Design Patterns. He is currently working as Associate Professor in the Department of Computer Science & Engineering of Prakasam Engineering College, Kandukur, Prakasam (district) A.P. India.