A Novel Technique to Improve the Performance of Licensed Software in Client/Server Network Environment

Dr. S. A. M. Rizvi  
Head Dept. of Computer Science  
Jamia Millia Islamia, New Delhi  
Email: samsam_rizvi@yahoo.com

Mr. Vineet Sharma  
Asst. Prof. Deptt. Of CSE  
KIET, Ghaziabad  
Email: vineet_sharma@kiet.edu

ABSTRACT
In the present era of ICT research and development, Resource Optimization has been one of the challenges for the performance of network environment. Moreover license softwares restrict and limit in terms of user/client machines due to their business requirements, which further becomes hurdle of the said resource optimization in the performance of network environment. In view of this there is a need to work out a strategy where one can improve the performance of the network environment through virtual extension with extra nodes and without breaching the license agreement. The present paper is an attempt to achieve the same in the well organized manner.

KEYWORDS
Application Service Providers (ASPs), Java 2 Enterprise Edition (J2EE), Business-to-business (B2B).

INTRODUCTION
Web technologies have taken a central place in current and future enterprise applications. Initially used for web publishing (i.e. building web sites through static HTML), web technologies have evolved into a “web tier” that is used as an integration glue between front-end and back-end applications and is at the foundation of current application platforms such as Microsoft’s .NET and Sun’s Java 2 Enterprise Edition (J2EE). Most current and future enterprise application are being developed and integrated by using web technologies, especially web services - a result of convergence of web with Distributed object technologies. In addition web technologies have enabled innovative business-to-business (B2B) trade and outsourcing models where application service providers (ASPs) can host a wide range of services just by providing a web interface at the customer site [1]. For example, a web service enabled payment system in Singapore can work with a web service enabled shopping cart in New York and a web service enabled inventory manager in London to support a global online purchasing system. With every software that is being developed on commercial scale, a license agreement is released along with it, defining the modes of usage of that software more specifically limiting to the number of nodes or users. Due to this agreement sometimes certain customer companies suffer, as there might be requirement of that software on some other system while the system on which originally the software was installed is not being used. This may hamper the overall performance of the company. To boost the performance of the network with in the given limitation of licensed user without indulging any illegality is a cumbersome exercise for the customer industry. Frequent increase of license user may not be always feasible to justify and purchase for the customer industry as well as could be difficult from the source and selling software industry due to administrative, financial and installation constrains.

Take a case where an organization purchases a database server license with 50 client machines software license. It means at most only 50 users can utilize the database server at any time. No more than that are authorized to use the database server; rather here the database server is optimized up to a maximum limit of 50 users, though mostly each and every client does not utilize the database server regularly. In this scenario database server is not loaded up to its optimum limit. Technique to utilize a database server among its client machines in more realistic manner. Now from the above points it is clear that for using any software we need to have its license key. To get it we need to purchase the software and then get the license key with the help of which we install the software on the machine. The problem which arises here is that with the introduction a new system we need to purchase the license key for it again which is done here provisionally. In this scenario an application is created with the help of which we can run the software on any new system. On the introduction of new systems in the company there is no need to purchase the license key, but remotely access the software from another system which is remotely located. According to this concept suppose in an organization a particular software is installed on 10 systems among which on 1 system the software is not used then with the help of application server other machines can remotely access the software.

WEB BASED SCENARIO
Web technologies were initially used for web publishing and advertising in enterprises. Technology-enabled business innovation presents the potential to structurally transform enterprise and industry practice, but uncertainty remains as to how such transformations might be managed [2]. However, over the year’s web technologies have been integrated with other distributed computing technologies and played a vital role in enterprise web architecture. Similar to the web evolution from static HTML to web-enabled business components that are at the foundation of the currently popular web services, this study presents a systematic and practical look at the utilization of the data-base server over LAN so that more numbers of
clients can utilize database without the requirement of extra client license. For example Google talk is based on Jabber protocol, works on the port no 5222 & uses XMPP (RF38920) protocol for sending data. For the time being we assume G-talk is not a free software. To use it we need to purchase a license key which is quite a tedious job. G-talk communicates with the server with the help of XMPP (Extensible Messaging and Presence protocol). The protocol is built to be extensible and other features such as voice over IP and file transfer signaling have been added. The logic which may be used in accessing the remote Google Talk software is as follow: There is a client server (my-server) which is running Google talk on the port no 5222 on its machine.

- The client server is communicating with the Google talk server via XMPP protocol & with the help of XMPP protocol it exchanges the data with the other client.
- Another client which is connected to the my-server has no Google talk installed on its local system.
- It sends the HTTP request to the server demanding what program is running on the port no 5222.
- In response it returns Google talk software, so it remotely access the exe file running on port no 5222 of my-server.

The history of web server-side programming in java started with servlets. Sun Microsystems introduced servlets in 1996 as small java based applications for adding dynamic contents to web application. Not much later with the increasing popularity of java, servlet took off to become one of the most popular technologies for internet development. Java Server Pages is another popular java technology for developing web application. JSP is an extension of the servlet technology. It is a common practice to use both servlet and JSP pages in the same web application. JSP uses the same techniques as those found in servlet programming. Servlet programmers know how cumbersome it is to program with servlet. On the other hand authoring JSP pages is so easy that any one can write JSP applications without much knowledge of its API.

Observing servlet execution, some disadvantages came into the focus. Whenever a web browser generates a request for a web page developed using servlet generally goGet() or doPost() method is invoked by the servlet container. These methods carry a stream of html tags and plain text. Each html tag must be embedded in a string and sent using the println() method of the PrintWriter class of the java.io package. It is a tedious job, worst still when the html page may be longer. Another disadvantage of using servlet is that every single change will require the intervention of the servlet programmer. Sun understood this problem and soon developed a solution. The result was JSP technology. Combining fixed or static template data with dynamic content is easier with JSP. JSP did not replace servlet as the technology for writing server-side internet/intranet application. In fact JSP was built on the servlet foundation and need the servlet technology to work.

**JSP FUNCTIONALITY**

Inside the JSP container is a special servlet called the Page compiler. The servlet container is configured to forward this page compiler all http requests with url's that matches the .jsp extension. This page compiler turns a servlet container into jsp container. When a .jsp page is first called, the page compiler parses and compiles the .jsp page into a servlet class. If the compilation is successful the jsp servlet class is loaded into memory. On the subsequent calls, the servlet class for that .jsp page is already in memory; however it could have been updated. Therefore the page compiler servlet will always compare the time stamp of the jsp servlet with the .jsp page. If the .jsp page is more current recompilation is necessary. With this process once deployed, jsp pages only go through the time consuming compilation process once.

**IMPLICIT OBJECT CASES**

Some objects are created by the jsp container and their references are automatically provided to all the jsp pages. These objects are called implicit objects. The following list shows the implicit object available with the jsp page with their respective classes.

<table>
<thead>
<tr>
<th>Object</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>request</td>
<td>javax.servlet.http.HttpServletRequest</td>
</tr>
<tr>
<td>response</td>
<td>javax.servlet.http.HttpServletResponse</td>
</tr>
<tr>
<td>out</td>
<td>javax.servlet.jsp.JspWriter</td>
</tr>
<tr>
<td>session</td>
<td>javax.servlet.HttpSession</td>
</tr>
<tr>
<td>application</td>
<td>javax.servlet.ServletContext</td>
</tr>
<tr>
<td>config</td>
<td>javax.servlet.ServletConfig</td>
</tr>
<tr>
<td>pageContext</td>
<td>javax.servlet.jsp.PageContext</td>
</tr>
<tr>
<td>page</td>
<td>javax.servlet.jsp.HttpServlet</td>
</tr>
<tr>
<td>exception</td>
<td>java.lang.Throwable</td>
</tr>
</tbody>
</table>

The javax.servlet.jsp package has two interfaces and four classes. The interfaces are as follows

- JSPPage
- HttpServletPage

The four classes are as follows

- JspEngineInfo
- JspFactory
- JspWriter
A Novel Technique to Improve the Performance of Licensed Software in Client/Server Network Environment

- PageContext

JspPage is the interface that must be implemented by all jsp servlet classes. The JspPage interface has two following methods
- public void jspInit ()
- public void jspDestroy ()

HttpJspPage directly extends the JspPage. There is only one method _jspService with the following signature

public void _jspService (HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

The Jsp container converts the jsp pages into their respective servlet classes. Following is the code, which is automatically provided with the jsp container.

JspFactory _jspFactory=JspFactory.getDefaultFactory ()
PageContext pageContext =_jspFactory.getPageContext (this, request, response,"", true, 8192, true);
ServletContext application=pageContext.getServletContext ();
ServletConfig config=pageContext.getServletConfig ();
HttpSession session=pageContext.getSession ();
JspWriter out=pageContext.getOut ();

JSP BEANS & COMPONENTS
The use of JSP components, such as java beans let the graphics designer do the page design and java programmer author the code. In this component centric approach the java programmer writes and compiles the java beans that incorporate all the functionality needed in the application. While the programmer is doing this, the page designer can work with the page designing at the same time. When the java bean is ready, the page designer uses tags similar to html to call methods and properties of the bean from the jsp page. Using beans in JSP application development is very much popular because java bean introduces reusability. This is to say rather than building a new piece of code, a person can simply use what other people have written. A bean is just a java class. There is no need to extend any base class or implement any interface. To be a bean however a java class must follow certain rules provided by the java bean specifications. In relation to java beans that can be used from a jsp page, some rules are as follows

- The bean class must have a no-argument constructor.
- The bean can have a public method that can be used to set the value of a property. This method is called setter method and it does not return any value. A setter method has the following signature:
  public void setPropertyName (PropertyType value)
- The bean can have a public method that can be called to obtain the value of a property. This method is called getter method and its return type is the same as the property type. A getter method has the following signature:

public PropertyType getPropertyName ()

To make a bean available on the JSP page jsp:useBean action element should be used. This element has attributes that can control the behavior of the bean. The two forms of the jsp:useBean action element are as follows:

<jsp:useBean (attribute="value")+   />

And

<jsp:useBean (attribute="value")+   >

initialization code

</jsp:useBean>
The (attribute="value") + means that one or more attributes must be present. The five attributes that can be used in a jsp:useBean action element are as follows:

- id
- class
- type
- scope
- beanName

Accessing property in a bean jsp:getProperty and jsp:setProperty action elements are used. The jsp:getProperty element is used to obtain the value of an internal variable and the bean must provide a getter method. The syntax of the jsp:getProperty is as follows:

<jsp:getProperty name=""Bean Name"" property=""propertyName""/>

The name attribute must be assigned the name of the bean instance from which the property value will be obtained. The jsp:setProperty action element is used to set the value of a property. Its syntax has following forms:

<jsp:setProperty name=""Bean Name"" property=""PropertyName"" value=""value""/>

<jsp:setProperty name=""Bean Name"" property=""PropertyName""/>

<jsp:setProperty name=""Bean Name"" property=""PropertyName"" param=""ParameterName""/>

<jsp:setProperty name=""Bean Name"" property=""*""/>

SESSION MANAGEMENT TECHNIQUES USING JSP
The first stage of the session management is where the web application containing JSP pages to form a connection pool which eventually connects to the concerned database server. These JSP pages demands the username and password of the database server in which the client wants to connect with the database. After checking the correctness of the username and the authentication of the password a new JSP page is forwarded to the client otherwise login-failed message is transferred back to the client. When the login process is successfully completed the client’s web browser gets a new JSP page where the user puts his/her sql query. After submitting this query page the page gets the result of the respective query on the same page. Here in this technique the client is not asked for the username and password repeatedly. Instead the submission of the query page every time sql query persists along with its result as provided by the licensed client software. This is done by providing the session management on the JSP pages. Session management can be done in web application by choosing any
of the following techniques: urlrewriting, cookies, hidden field and by session management. In the proposed technique java beans are used in the jsp pages. Client’s information is saved in form of property of the java bean used in the JSP page. N different users can maintain their separate information with the help of java beans. The following snip works in this scenario.

```java
public class SQLToolBean {
    private String sql = ""
    private String userName = ""
    private String password = ""
    private String connectionUrl;

    public SQLToolBean() {} 

    public String getSql() { return(sql); }
    public void setSql(String userName) {
        if (sql!=null)
            this.sql=sql;
    }
    public void setUserName(String userName) {
        if(userName!=null)
            this.userName=userName;
    }
    public String getUserName() {
        return userName;
    }
    public void setPassword(String password) {
        if(password!=null)
            this.password=password;
    }
    public String getPassword() {
        return password;
    }
    public void setConnectionUrl(String url) {
        connectionUrl=url;
    }
    public String getResult() {
        if(sql==null||sql.equals(""))
            return "";
        StringBuffer result=new StringBuffer(1024);
        try {
            Connection con=DriverManager.getConnection(connectionUrl,
            userName, password);
            Statement st=con.createStatement();
            if(sql.toLowerCase().startsWith("SELECT"))
                result.append("<TABLE BORDER=1>");
                for (int i=1; i<=columnCount; i++)
                    result.append("<TD><B>"+rsmd.getColumnName(i) +"</B></TD>
                result.append("</TR>");
                while (rs.next())
                    result.append("</TR>");
                for (int i=1; i<=columnCount; i++)
                    result.append("<TD">+rs.getString(i)+"</TD>");
                    result.append("</TR>");
                rs.close();
                result.append("</TABLE>");
            else
                int i=st.executeUpdate(sql);
                result.append("Record(s) affected: "+i);
            st.close();
            con.close();
            result.append("</TABLE>");
        }
    }catch(Exception e) {
        result.append("<B>Error</B><BR>"+e.toString());
    }
    return result.toString();
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
version="2.4">
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>
            index.jsp
        </welcome-file>
    </welcome-file-list>
</web-app>
```

```jsp
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
```
A Novel Technique to Improve the Performance of Licensed Software in Client/Server Network Environment

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<% @page language="java" contentType="text/html;charset=UTF-8">
<title>My Sql Tool.....</title>
<body>
   <%
      try {
         Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      }
      catch(Exception e){out.println(e.toString());}
   %>
   <jsp:useBean id="theBean" class="vin vin1.SQLToolBean">
   <jsp:setProperty name="theBean" property="userName"/>
   <jsp:setProperty name="theBean" property="password"/>
   <jsp:setProperty name="theBean" property="connectionUrl" value="jdbc:odbc:vin"/>
   <jsp:setProperty name="theBean" property="sql"/>

<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>My Sql Tool....</title>
   </head>
   <body>
      <c:if test="${param.sayHello}">
         <!-- Let's welcome the user ${param.name} -->
         Hello ${param.name}!
      </c:if>
      <BR><H2>My Sql Tool</H2>
      <BR>Please Type your sql statement in the following box.
      <BR>
      <Form method=post>
      <input type=hidden name=userName value="<jsp:getProperty name="theBean" property="userName"/>">
      <input type=hidden name=password value="<jsp:getProperty name="theBean" property="password"/>">
      <textarea name=sql cols=80 rows=8>
      <jsp:getProperty name="theBean" property="sql"/>
      </textarea>
      <BR>
      <INPUT TYPE=SUBMIT>
      </Form>
      <hr>
      <hr><br>
      <%=theBean.getResult()%>
   </body>
</html>

PERFORMANCE & EFFICIENCY MEASURES
The application server is solely responsible for hosting, configuring and for the execution of the web application. This web application maintains a connection pool with the database server through the jdbc drivers. Following are the popular jdbc driver available with us:
- Jdbc odbc bridge drivers
- Native API drivers
- Native Protocol drivers
- Network Protocol drivers

In the proposed web application pure java thin drivers are being used. These drivers are provided by the database itself. Using the database with the help of these drivers adds one more additional layer in between the client and the database server. This means, using efficient jdbc drivers in JSP pages in the web application we are still little bit slower as compared to those licensed client software who are connected database server directly. However on the other hand this technique provides the availability of database server among more number of clients as directed by the agreement of client software license and that too by the “Distribution and usage of software over a local area network without actually installing it or saving it on any node”.

CONCLUSION
The present study is an attempt to improve the management of software resources in an organized manner with in a network environment by adoption of potentially disruptive future generation web technologies. Not only the effective utilization of database, using and implementing the same approach, where one can be able to provide the various types of software installed on some specific machines to the other machines of the same network. Application Service Providers (ASPs) can be equipped and registered with various software and can provide them to the other machines on demand. It therefore improves and enhances the performance of the whole network environment with higher degree of optimization of resources without the breaching of license software.

FUTURE SCOPE
The concept mentioned in this study highlights the usage of single user licensed software on various computer systems. This concept can be extended to the mobile softwares as well. There might be certain other complex challenges involved in the case of implementing it in mobile technology. This will prove to be a very beneficial to mobiles which contains limited resources in terms of memory, processing power, structure etc. On demand software could be a next step in mobile technology, where a mobile user can request any software from the network provider whenever required and use it on his/her mobile phones without actually installing the software on the mobile phone.

REFERENCES
[2] Steve Elliot and Mary-Anne Williams and Niels Bjorn-Anderson “Strategic Management of technology-enabled disruptive innovation: Next Generation Web


