Rasterization of Fragmented Spatial Data

Dahlberg Marina

January 14, 2014
Master’s Thesis in Computing Science, 30 credits
Supervisor: Jerry Eriksson
Examiner: Fredrik Georgsson

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

The Geographical Information Systems (GIS) is nowadays a large industry that has evolved from a highly specialized niche to a technology that affects nearly every aspect of our lives. There is a big challenge to use the functionality of GIS within the organization that works with data in the ordinary old-fashioned way using files stored locally in the computer. The availability of sharing and visualizing data forces an organization to invest in modern software solutions. Sweco is one of the organizations which offer the software solution SMIL to complement the information stored in the organization with spatial support.

The aim of the work presented in this thesis was to measure the time needed for rasterization of an image map with different amount of features in a simplified prototype of SMIL with similar data flow organization. This prototype was developed in consultation with Sweco's software architect and GIS consultant and was tested using the organization's network capacity.

Four different types of tests, which were implemented in order to investigate the presence of possible tipping points, illustrated the similar result that when the number of requested features passes one thousand, both the time needed for rasterization and the size of the raster image increases rapidly. The fifth test, that was implemented in order to analyze the time the involved modules in the system needed to generate a response, identified GeoServer as an apparent critical module in the system that delays data flow when the number of requested features passes one thousand and it can slow down the system when the number of requested features passes ten thousand.

Contents

List of Figures

List of Tables

… # Acknowledgements

I would like to thank my GIS supervisor Peter, Sweco architect Per and GeoServer expert Henrik who have helped me during this work by answering my questions and generously giving me their time and sharing their opinions and ideas about the designed system. I would like to thank my teacher by giving me constructive feedback and helping me to structure this report.

1. Introduction

The first part of this thesis is an introduction to the goal and purpose of the project with some explanation about the requirements and the background information about the organization that was interested in this project. The second part is a theoretical part which presents relevant research and explains theoretical background needed to understand elements of the system that was implemented and tested in the project. The third part describes the design and implementation of the system and it starts with some brief introduction into existing system SMIL with more complex functionality, which was built by the Sweco Position in order to communicate with the Share Point platform. The system implemented in this project was built as a much more simplified version of SMIL in order to simulate the storage and the organization of data flow. The fourth part presents different types of tests that were implemented in order to investigate tipping points and possible bottlenecks in the system. The fifth part describes the problems that occurred during implementation of the system, how these problems were solved and what type of limitations the system has. The last part is a conclusion part where important experience and analyzes is drawn. References, source and TimeLog are attached in the end of this thesis. A list of figures and a list of tables are attached after Contents of this thesis. Examples with source are highlighted with different font.

1.1 Overview

There is a big challenge within GIS to use the functionalities such as analyzing, storing, sharing and visualizing of data in the organization that works with data in the ordinary old-fashioned way using the copies of text or excel files stored locally in the computer. The availability of sharing data and visualizing it on the map forces organizations to invest in modern software solutions. Sweco is one of the organizations which offer the possibility to complement the information stored in the organization with spatial support.

1.2 Goals and purpose

The goal of the project is to investigate the possibilities to the rasterization of the fragmented spatial data between GIS and Microsoft Share Point platform.

The purpose of the project is to measure and analyze the time during rasterization of the fragmented spatial data by the implementing a much more simplified version with similar organization of data storage and data flow as in the existing system SMIL. The performance should be measured for one hundred, one thousand, ten thousand and one hundred thousand features, such as points, lines and polygons. The first milestone is to identify any tipping points during rasterization in the implemented system. The second milestone is to identify any possible bottlenecks if the occurrence of tipping points was observed. The relevant performance should be discussed with Sweco Software Architect.

1.3 Methods

The identifying of tipping points was done by implementing four tests: “Test one layer”, “Slice test”, “Test one zoom” and “Test all layers”. All these tests were designed with respect to how a common user interacts with a map. The identifying of any possible bottlenecks was done by implementing a test that measured the time between components in the implemented system.

1.4 Problem statement

During the planning and discussion about the project, one trade-off was done in order to scale down the implementation of the simplified version of the existing system. This trade-off implied exclusion of using Share Point platform in the project because of the fact that the only functionality of Share Point platform and not the Share Point itself was of interest. The discussion resulted in the agreement that the implemented version should simulate the demanded functionality of the Share Point without using this software in the project.

2. Related work and theoretical background

This part of the thesis consists of an introduction to the related work described in section [2.1] and the theoretical background to the components of the project presented in [2.2], which define and explain the framework for understanding the implemented system.

2.1 Related work

The project includes a set of different network and software components widely studied and discussed in the literature. This part of the chapter summarizes the works which found to be relevant for the project.

GIS

Geographical Information Systems (GIS) is often described as integration of data, people, hardware and software designed for management, processing, analyzing and visualization of geographically referenced information. The current GIS technology spans a wide range of applications from viewing map and images on the web to spatial analysis, modeling, and simulations. [10]

GIS applications are used in several areas such as environmental systems, transportation systems, emergency response systems and battle management. Besides the widely used proprietary systems, there exists an Open Source GIS as for example GRASS (Geographical Resources Analysis Support Systems) , that is created and supported by Open Source Geospatial Foundation (OSGeo[1]) in order to provide access to GIS for the users who cannot or do not want to use proprietary products. [10]. More theoretical information about GIS is found in section [2.2.1]

WebGIS

The general problem of retrieval and integration of spatial data from a distributed heterogeneous data sources discussed by M. Howard Williams and Omar Dreza in [8] is a continued research of two related problems: the retrieving and integrating complexity problem started by El Khatib [8], and the breakdown problem of a query into appropriate sub-queries that can be applied to different data sources, introduced by MacKinnon [8].

Wrapper problem whose purpose is to translate a query from the server language into the language that is understandable by the Relational Database Manager and transform the result received from the data source to the server language is discussed by Zaslavsky [8].

[1] http://osgeo.org

WMS

Web Map Service (WMS) has a long history, which started with description of "WWW Mapping Framework" by A. Doyle in [2]. There was the first Web Mapping document within the Open Geospatial Consortium (OGC). A. Cuthbert in his work "User Interaction with Geospatial Data" [1] defined the first OGC consensus position of the WWW Mapping Special Interest Group, which is the core task force of OGC. From these two documents, as well as from "A Web Mapping Scenario", the OGC initiative known as the Web Mapping Testbed (WMT) was begun. [25]

That initiative was first described in a Request For Technology (RFT) [11] and then in the Request for Quotation (RFQ) [12]. Web Mapping Testbed had two phases: the first phase supported only basic interoperability of simple map servers and clients culminated in the Web Map Service Interface Implementation Specification, "WMS 1.0.0". During the phase 2 Web Mapping Testbed was developed with more advanced features and culminated in WMS 1.1.0 and later in WMS 1.1.1. This version WMS 1.1.1 is used in the project. For more information about WMS see section [2.2.7].

REST

The term Representational State Transfer (REST) was introduced and defined in 2000 by Roy Thomas Fielding in his dissertation "Architectural Styles and the Design of Network-based Software Architectures" [5] as an attempt to understand and evaluate the architectural design of network-based application software architecture via architectural styles . For more information about REST see section [2.2.8]

To sum up, the progress in network infrastructures to distribute geospatial information, the policies and possibilities of sharing of geospatial data between municipality and other organization, the software architectures that provide interactive GIS functionality, the database technologies that facilitate distribution of spatial data, these all are the standpoints that keep the interest to the problem of retrieval, integration and distribution of geospatial data.

2.2 Theoretical background to the components of the project

This part of the thesis defines the framework for understanding the implemented system. Each software component or technology are defined and presented separately in the appropriate section.

2.2.1 GIS

The georeferenced data is the core of GIS applications which provide a simplified representation of Earth features for a given region and include a spatial component (called Spatial data) that describes the location or spatial distribution of geographic phenomenon and an attribute component (called Attribute data) used to describe its properties, see [Figure 1: Geographically referenced information about a property]

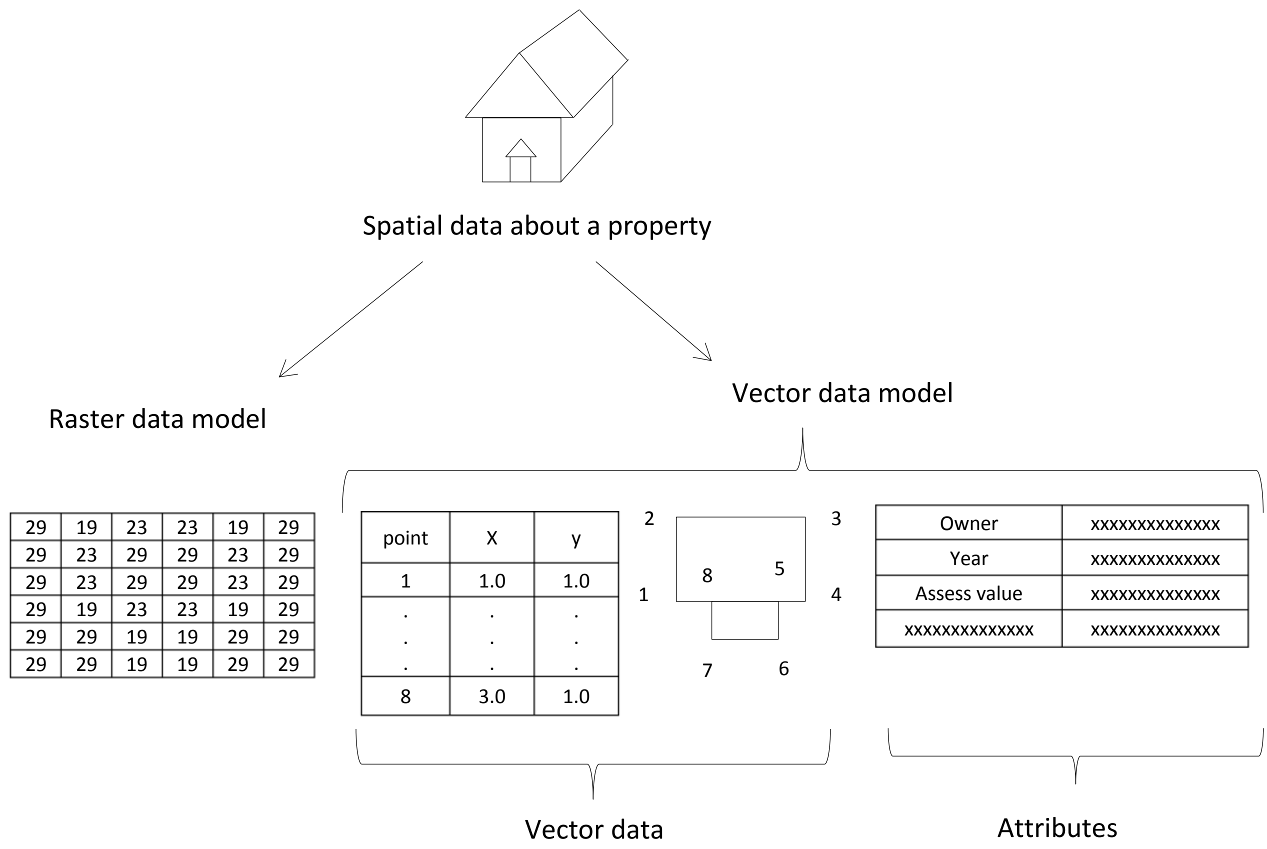


Figure 1: Geographically referenced information about a property

Spatial data can be obtained from satellite images, scanned maps or other resources, then digitized and represented using one of two approaches: raster data model where each pixel has an assigned value or vector data model where geographic features are defined as points, lines, and polygons given by their coordinates, see:[Figure 1: Geographically referenced information about a property]. There are two types of coordinate systems: geographic coordinate systems, which use latitude and longitude as angles measured from the earth's center (called datum) and projected coordinate systems, which use a projection method to project coordinates from the earth's spherical surface onto a two-dimensional Cartesian coordinate plane. There are several different projections developed by cartographers and mathematicians, but there is no best projection, hence each projection modifies the data and includes some deformations about length, areas or shapes. The information about projection and the Spatial Reference System (SRS) is stored in Spatial Reference Identifier (SRID) using the Open Geospatial Consortium's (OGS) well-known text (WKT) representation. The SRS for the geographic

WGS84 reference system used in the project is presented in [Table 1: Spatial Reference System WGS84].

Spatial Reference System WGS84	
SRID	WKT representation

EPSG:4326	`GEOGCS["WGS 84",`
	`  DATUM["World Geodetic System 1984",`
	`    SPHEROID["WGS 84", 6378137.0, 298.257223563,`
	`              AUTHORITY["EPSG","7030"]],`
	`    AUTHORITY["EPSG","6326"]],`
	`  PRIMEM["Greenwich", 0.0, AUTHORITY["EPSG","8901"]],`
	`  UNIT["degree", 0.017453292519943295],`
	`  AXIS["Geodetic longitude", EAST],`
	`  AXIS["Geodetic latitude", NORTH],`
	`  AUTHORITY["EPSG","4326"]]`

Table 1: Spatial Reference System WGS84

Spatial data can be re-projected from one coordinate system into another, which implies the possibility to integrate data from various sources using GIS software.

Attribute data is the detailed data also called descriptive data associated with the spatial data. This data can be obtained from a number of sources such as town planning, management departments, policing, fire department or online media. Attributes are usually managed by external or internal GIS database management systems (DBMS) using corresponding coordinates or identification numbers to link the attributes to the geometric data. Both spatial data and attributes have to be in the same coordinate system in order to be layered together for mapping and analysis. Some database management systems extender, such as PostGIS allow the user to store spatial data into the database, for more information about storing geospatial data se section [2.2.4].

2.2.2 Shapefile

This part of thesis provides the important information about structure of a shapefile and how the geometry of a feature is stored in such a file.

A shapefile stores nontopological geometry and attribute information for the spatial features in a data set [4]. Shapefiles can support point, line and area features which are represented as closed loop, double-digitized polygons. The geometry for a feature is stored as a shape comprising a set of vector coordinates and each attribute record, which is stored in a dBASE® format file has a one-to-one relationship with the associated shape record. An ESRI[2] shapefile

[2] www.esri.se/

consists of a main file, an index file, and a dBASE table. The content of a shapefile are summarized in [Table 2: ESRI shapefile].

Type of file	Extension	Description
Main file	.shp	The main file contains a fixed-length file header followed by variable-length records in which each record describes a shape with a list of its vertices.
Index file	.shx	Each record in the index file contains the offset of the corresponding main file record from the beginning of the main file.
dBASE table	.dbf	The dBASE table contains features attributes with one record per feature, where one-to-one relationship between geometry and attributes is based on record number. Attribute records in the dBASE file must be in the same order as records in the main file.

Table 2: ESRI shapefile

Geometry	Description	example
Point	A point consists of a pair of double-coordinates X,Y.	`Point { Double X Double Y }`
PolyLine	A PolyLine is an ordered set of vertices that consists of one or more parts, where a part is a connected sequence of two or more points. Parts may or may not be connected to one another, and they may or may not intersect one another.	`PolyLine { Double[4] Box Integer NumParts Integer NumParts Integer[NumParts] Parts Point[NumPoints] Points }`
Polygon	A polygon consists of one or more rings, where each ring is a connected sequence of four or more points that form a closed non-self-intersecting loop. A polygon may contain multiple outer rings. Vertices of rings that define holes in polygons are in counterclockwise direction. Vertices for a single ringed polygon are always in clockwise order.	`Polygon { Double[4] Box Integer NumParts Integer NumParts Integer[NumParts] Parts Point[NumPoints] Points }`

Table 3: Geometry representation in a shapefile

All the contents in a shapefile can be divided into two categories: data related and file management related [4]. The brief description of how point, line and polygon are represented in a shapefile record content is summarized in the [Table 3: Geometry representation in a shapefile]. Information about how to read a shapefile is found in section [3.4.1].

Because shapefiles do not have the processing overhead of a topological data structure, they require less desk space and are easier to read and write. Shapefile format have advantages over other data sources depending on faster drawing speed and edit ability, and that is why this file format is widely used in GIS.

2.2.3 GeoServer and Apache Tomcat

This part of the thesis provides the important information about GeoServer installation and how it works with a spatial data.

GeoServer is a Java web application that needs Java Runtime Environment (JRE) in order to run the application, a servlet container on top of the JVM that implements Java servlet and JavaServer Pages technologies and is responsible for managing the lifecycle of servlets, mapping a URL to a particular servlet, access security, and optionally Java Development Kit (JDK) in order to compile Java™ code, while developing the GeoServer [9]. Because Apache Tomcat, as an open source project of Apache foundation, is widely adopted in the GeoServer developer's community and well-documented, this servlet container was installed from [27] and used in the project.

GeoServer Web Archive version 2.3.5 was downloaded from [26]. The war file for GeoServer is bigger than what Tomcat 7 Manager has as default limit for deployable application, therefore the `max-file-size` and the `max-request-size` in `$CATALINA_HOME/webapps/manager/WEB-INF/web.xml` should be set to a safe size for GeoServer, set to 62914560 (60MB).

```
<multipart-config>
   <!- - 50MB max - - >
   <max-file-size>62914560</max-file-size>
   <max-request-size>62914560</max-request-size>
   <file-size-threshold>0</file-size-threshold>
</multipart-config>
```

For more detailed information about deploying GeoServer on Tomcat see Chapter 2 in [9]. GeoServer is managed from an administrative interface, see [Figure 2: GeoServer administrative interface].

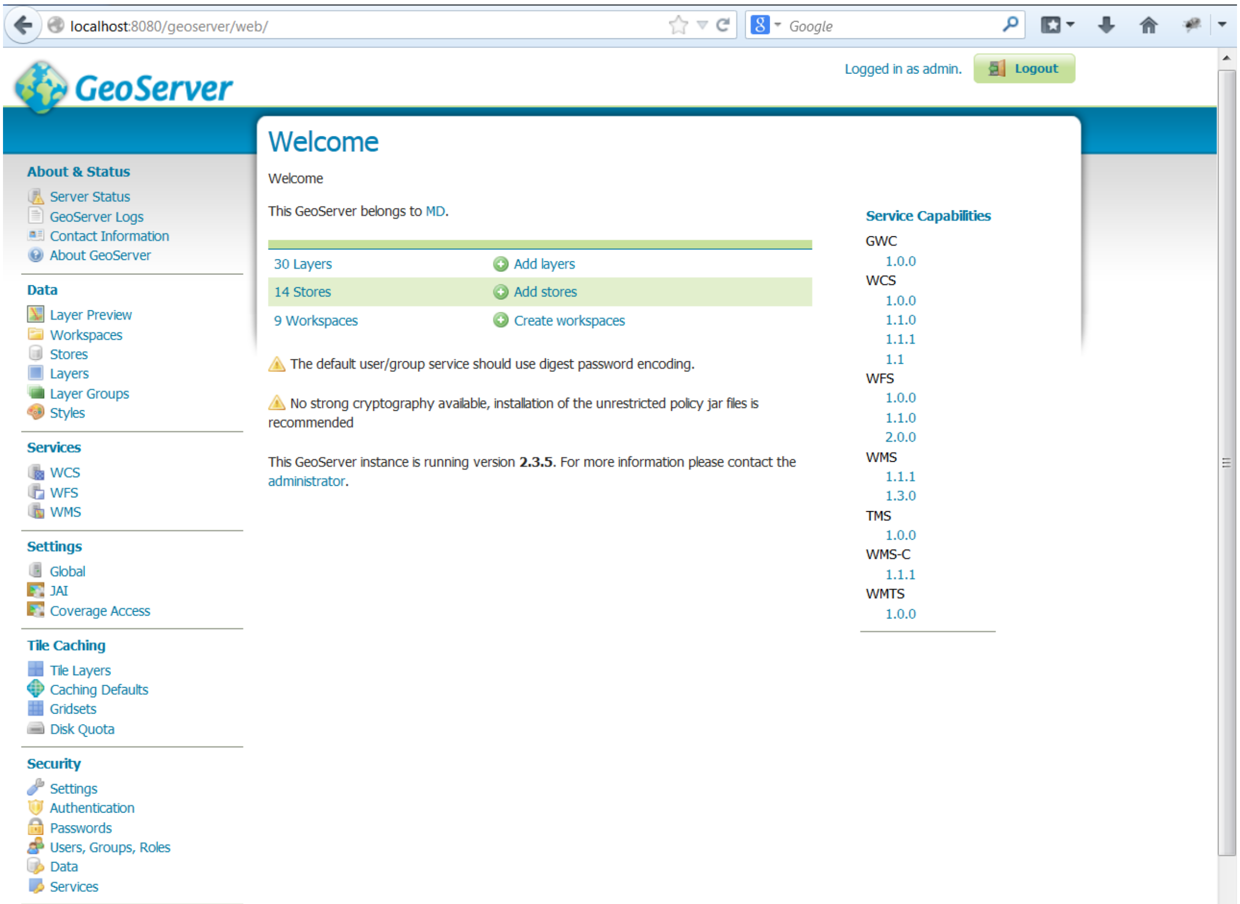


Figure 2: GeoServer administrative interface

On the left-hand-side there is a table of contents with administrative operations available in the GeoServer. The section called Data includes all functionality needed to work with spatial data and to configure the data access. Layer Preview lists every layer with features known to GeoServer, Workspaces is useful for organizing layers, Stores let GeoServer know where the spatial data is and what it is, Layers get a direct access to the specific layer and Styles help to visualize feature in the layer. On the right-hand-side there is a list with all possible GeoServer capabilities. The WMS 1.1.1 was used in this project.

Information about how a layer with a given type of features is published in the GeoServer is found in section [3.4.2].

2.2.4 PostgreSQL and PostGIS

This part of the thesis provides the information about how a relational database becomes a spatial database and how a spatial database stores and manages a spatial data.

PostGIS is extension to the PostgreSQL object-relational database system which allows store GIS object in the database and includes support for a range of important GIS functionality such as: GiST-based R-Tree spatial indexes, advanced topological constructs, functions for analyzing geometric components, determining spatial relationship, manipulating geometries and processing of GIS objects [14].

Adding PostGIS turns the PostgreSQL Database Management System into a spatial database where spatial features are treated as first class database objects and spatial data is fully integrated with an object relational database [15].

The main difference between the relational database and the spatial database is the way the databases store and process data. Whereas relational database store and process numeric and character data, the spatial database store spatial data types which are organized in a type hierarchy [Figure 3: Geometry Hierarchy] [15] where each subtype inherits the structure (attributes) and behavior (methods or functions) of its super-type. Spatial structures such boundary and dimension are abstracted and encapsulated within a data type.

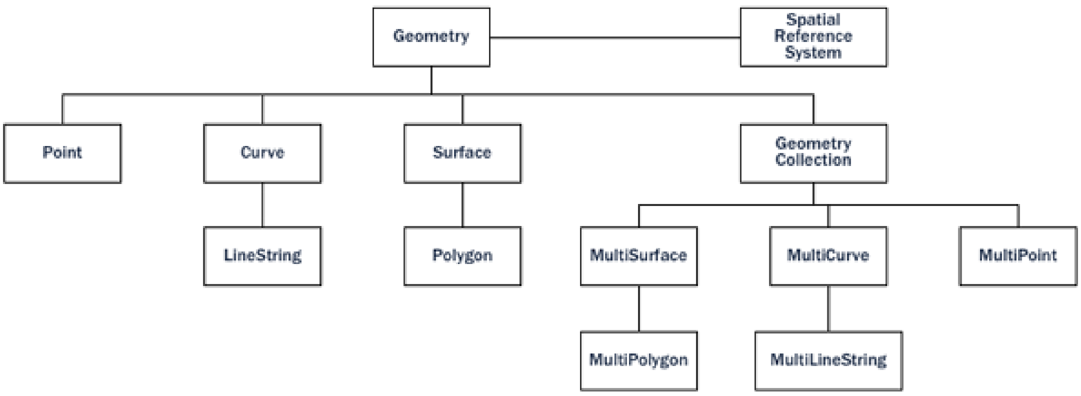


Figure 3: Geometry Hierarchy

A spatial database is optimized to store and process queries with spatial parameters, also called spatial queries, related to topological relationship among objects in space, including points, lines and polygons [3].

Information about how to create a table to store the spatial data and how to read the spatial data into such a table is presented in section [3.4.1].

2.2.5 SQLExpress

SQL Server Express 2012 is a full-featured relational database management system (RDBMS) developed by Microsoft that includes a variety of administrative tools, such as SQL Server Management Studio, Configurations and Performance Tools, Integration Services and Analysis Services. Books Online for SQL Server and Server Technologies are found on the Microsoft website [22]. This database management system is used to store attribute data by collaborating with SharePoint platform in the organization and in the project this RDBMS was used in order to simulate data flow correctly, see section [3.4.1]

2.2.6 OpenLayers 2.10

"OpenLayers is a client side JavaScript library for making viewable interactive web maps in nearly any web browser." [7] Originally this JavaScript library was developed by Metacarta, as a response to Google Maps. OpenLayers operates according to Client/Server model, where map client communicate with a web map server, such as a WMS server or Google Maps backend, in order to get a map images, see [Figure 4: Client / Server model].

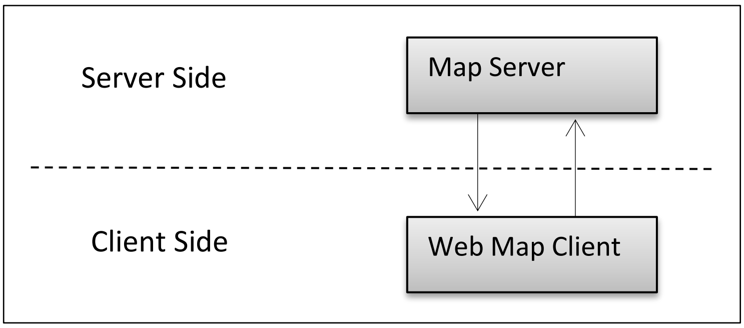


Figure 4: Client / Server model

The OpenLayers API (Application Programmer Interface) can be stored locally or linked to a JavaScript file served on the site.

```
<script src="OpenLayers.js" type="text/javascript"></script>
<script src="http://openlayers.org/api/OpenLayers.js"
          type="text/javascript"></script>
```

OpenLayers allows using and combining a set of different server backends (also called as map server or map service) such as WMS, Google Maps, Yahoo! Maps, ESRI ArcGIS, WFS, Open Street Map on the same map by creating an appropriate layer object and then adding it to the map. The general rules of creating a layer object presents in the [Table 4: Layer WMS class].

Each time a user navigates or zooms around on the map, the client sends new asynchronous JavaScript (AJAX) request to the map server for map images and puts the returned map images together by using OpenLayers API.

Parameters	Description
name	{String} A name for the layer
url	{String} Base url for the WMS
params	{Object}An object with key/value pairs representing the GetMap query string parameters and parameter values
options	{Object} Hashtable of extra options to tag onto the layer
example:	`var wms_layer = new OpenLayers.Layer.WMS( "Base layer", "http://vmap0.tiles.osgeo.org/wms/vmap0", {layers: 'basic'}, {isBaseLayer: true} );`

Table 4: Layer WMS class

For more information about WMS layer see documentation for the WMS class at [13].

WMS layer in the project contains `url` to the locally stored GeoServer, see section [3.3].

2.2.7 WMS

Web Map Service (WMS) produces maps of specified georeferenced data. Open GIS Consortium Incorporation defines concept of "map" as a visual representation of geodata and emphasizes that a map is not a data itself [25]. There are three WMS operations that are important to know before using WMS. The first operation is *GetCapabilities* operation that returns service-level metadata, such as a description of the service's information content and what type of parameters in a request are acceptable by WMS. The second operation is *GetMap* operation that returns a map image whose geospatial and dimensional parameters are well defined [Table 5: A general OGC Web Service Request]. The third WMS operation *GetFeatureInfo* is an optional operation that returns information about particular features shown on a map.

All these operations can be invoked by using World Wide Web (WWW) Uniform Resource Locators (URLs) prefix to which additional parameters are appended in order to construct a valid request. URL prefix should include the protocol, hostname, optional port number, path, a question mark '?', and one or more server specific parameters separated by '&'.

The basic idea behind requesting a map is that a client sends a request which specifies the information to be shown on the map. This request usually includes one or more layers, possibly styles of those layers, what portion of the Earth is of the interest (Bounding Box), which coordinate reference system to be used: projected or geographic, the desired output format (GIF, PNG etc.), size (Width and Height), background transparency and color.

<table>
<tr><td>URL Component</td><td>Description</td></tr>
<tr><td><code>http://host[:port]/path?{name[=value]&}</code></td><td>[] denotes 0 or 1 occurrence of an optional part
{} denotes 0 or more occurrences</td></tr>
<tr><td><code>name=value&</code></td><td>Parameter name/value pairs defined by an OGC Web Service.</td></tr>
<tr><td colspan="2"><pre>Example name/value pairs:

            request=GetMap&
            srs=EPSG:4326&
            service=WMS&version=1.1.0&

Example url:

http://localhost:8080/geoserver/MD/wms?service=WMS&version=1.1.0&request=GetMap&lay
ers=MD:points_100&styles=&bbox=380000.0,7000049.0,382000.0,7000051.0&width=2970&hei
ght=330&srs=EPSG:4326&format=image%2Fpng</pre></td></tr>
</table>

Table 5: A general OGC Web Service Request

Map layers can be requested from different Servers and when two or more maps are produced using the same Bounding Box, Spatial Reference System, output size, and transparent backgrounds, the result can be layered on the client side producing a composite map.

2.2.8 REST

REST is a hybrid architectural style derived from several existing network-based architectural styles and combined with additional set of architectural constraints for connecting the Internet-scale distributed hypermedia system. This architectural style was developed by Fielding using the following process of architectural design approach: a designer starts with the system needs without any constraints. Constraints are identified and applied to elements of the system incrementally in order to differentiate the design space and to allow the forces that influence the system behavior to flow naturally [5].

The starting point for REST was a system without distinguished boundaries between components. By adding first client-server, then stateless and then cache constraints the system induced the properties of visibility, reliability and scalability; such that each request from client to server contains all necessary information for understanding this specific request. At this point the designed architecture guarantees that the request cannot take advantage of any stored context on the server, and if response is cacheable, then a client cache is given the right to reuse that response data for later equivalent requests. [5] The constraint of using uniform interface between components is the central feature that distinguishes REST from other network-based styles. Combination with layered system constraint improves behavior for Internet-scale requirements, because hierarchical layers can be used to encapsulate and protect components during interaction. Code-on-demand is an optional constraint which allows downloading and executing code in form of applets or scripts, which consequently reduce the number of features to be pre-implemented on the client side [Figure 5: REST by Fielding Roy Thomas].

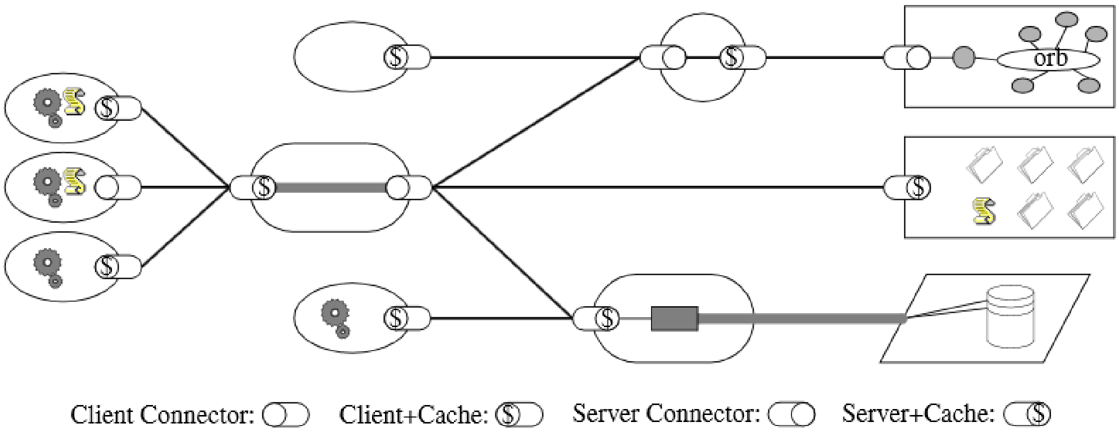


Figure 5: REST by Fielding Roy Thomas

When constraints in REST are applied as a whole, this architectural style emphasizes scalability of component interactions, generality of interfaces, independent deployment of components, and intermediary components to reduce interaction latency, enforce security and encapsulate legacy systems. [5] Within REST the components can actively transform the content of self-descriptive messages which semantics are visible to the intermediaries.

2.2.9 REST with ASP.NET

REST within ASP.NET is a two-way data flow interaction, in which clients use URLs and HTTP operations GET, PUT, DELETE and POST in order to manipulate resources that are represented in XML. You gain low-level access to HTTP request and response by using an HTTP handler, which handles all requests made for a file with a certain extension, path or request type. [24] ASP.NET includes several built-in HTTP handlers:

- ASP.NET page handler (*.aspx) is a default HTTP handler for all ASP.NET
- Generic Web Handler (*.ashx) is a default HTTP handler for all Web handlers that do not have a UI and that include @WebHandler directive, such as <%@WebHandler attribute = “value” [attribute = “value”...]%>
- Web Sesrvice handler (*.asmx) is a default HTTP handler for web service pages created as .asmx files in ASP.NET
- Trace handler (trace.axd) is a handler that displays the current page trace information

A request to an ASP.NET is mapped by the PageHandlerFactory class to an appropriate HTTP handler based on a file name extension in order to service the request.

There are two steps to be done to create an HTTP handler:

1. Create a class that implements the IHttpHandler interface. This step requires you to implement one property: IsReusable, which indicates whether the current handler can be reused for another request and one method `ProcessRequest()`, which contains the actual code to be executed in response to the request.
2. You have to add a reference to the HTTP handler in the Web.Config file to associate the handler with a set of pages requested in current directory and all its subdirectories.

When a specific HTTP handler is requested, ASP.NET calls the `ProcessRequest()` method of this handler, which process the request, creates response and sends it back.

2.2.10 ASP.NET and Visual Studio

ASP.NET Web Application projects runs by default by using the built-in Visual Studio Development Server. When you run the application, Visual Studio compiles the project into a single assembly. When you debug the application by pressing Ctrl+F5, Visual Studio attaches a debugger to the Web Server Process. All project settings are saved after that in Microsoft Build Engine project file.

3 Implementation of the environment

This part of the thesis presents the architecture of the system that was developed and implemented in order to measure time needed to service a request and screen the fragmented spatial data. It starts with a short presentation of some functionality of the Share Point, for more details see section [3.1] and how Share Point platform interacts with SMIL, see section [3.2]. Design considerations about the system and its components are presented in [3.3]. Information about implementation of each component is found in sections [3.4] through [3.4.3].

3.1 SharePoint

This part starts with an overview presentation of how information is organized and stored on the Share Point collaboration platform, which allows teams to manage, store and share documentation. Information about different versions of SharePoint, tutorials and other documentation is available on Microsoft website [17] and [18].

SharePoint can be described as a collection of Web Sites, where a site may be created for entire organization, or for just one document. Information that is found on SharePoint Site is stored in Lists, which are a key part of the architecture of Windows SharePoint Services [19].

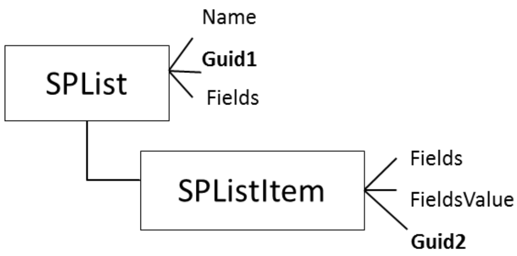


Figure 6: SPList

A list consists of items or rows, and columns or fields that contain data. [20] Each List has its own Globally Unique Identifier, Guid1 in [Figure 6: SPList] and each item that belongs to the list is coupled to that identifier and has its own Guid2. For information about how this was implemented in the project see section [3.4.1].

3.2 SMIL

SMIL is a software system developed by Sweco Position that supplies the information stored in SharePoint with spatial support, which results in the possibility to visualize this information by putting it on the map. SMIL has a wide range of functionality, compatible with widely used software, and is portable with mobile devices, see [Figure 7: System overview SMIL] made by Sweco Position about SMIL.

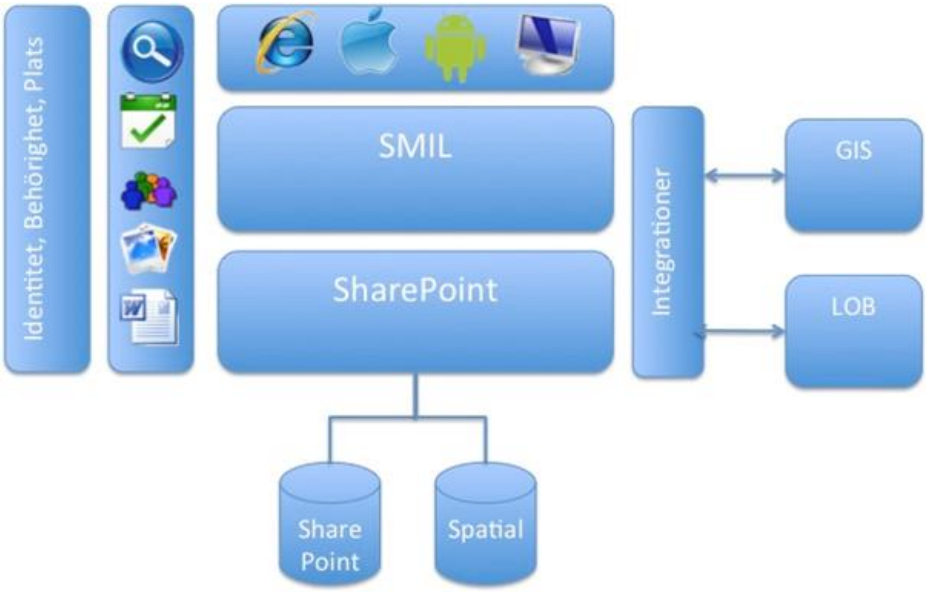


Figure 7: System overview SMIL

SMIL can be used with drawing archive. By using SMIL pictures that are placed on the map obtain the appropriate attribute and spatial data. SMIL mobile map client can be used with Android, iPad to access information stored in Microsoft SharePoint by WFS service.

3.3 Design

The architecture developed and implemented in this project is a simplified version of the existing architecture SMIL, briefly described in section [3.2], with focusing on functionality to support access from the SharePoint platform to the distributed resources.

First simplification of SMIL excludes SharePoint platform and simulates a part of its functionality by a module Service.ashx, which supports access to three distributed resources: a collection of heterogeneous spatial data stored in PostgreSQL database, a reduced collection of attributes related to the spatial data stored in SQLExpress database and the GeoServer that produces the information. The goal of the second simplification is to exclude network latency by placing resources on the same computer.

The architecture is based on a client/server approach with three levels: the client level, the server level and the data provider level. The basic idea behind this architecture is that the client should be able to send a request with a query which requires geospatial data placed in different sources and receive the response produced by the system without being aware of the different sources involved. A system based on this architecture was implemented using C#.

The overview of the design behind the developed system can be described by a [Figure 8 System architecture], where client level is presented by Client module, server level is built-up of two different servers: GeoServer and built-in Visual Studio Development Server, and data provider level is presented by two Database Management Systems: SQLExpress and PostgreSQL. For more detailed information about implementation of each part see section [3.4].

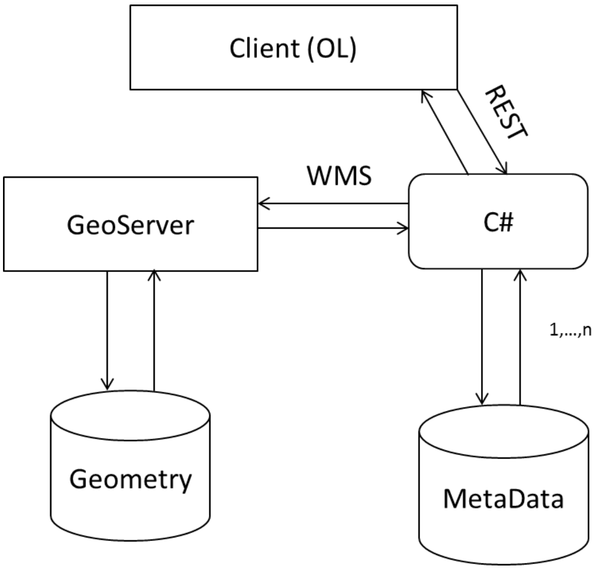


Figure 8 System architecture

By applying client / server approach and by separating server level from the data storage level, the portability across different software platforms and the scalability of the designed system improves.

Since the main focus of this project was on the measurements this provides a simple and easy to use GUI limited testbed for this specific purpose. It is worth to point that all tests were observed by using a web development tool Firebug version 1.12.5 with the Firefox browser version 26.0.

3.4 Implementation

This part of thesis presents detailed information about implementation of the designed system described in section [3.3] and visualized by [Figure 8 System architecture]. Figures [Figure 9: Import into DBMS] through [Figure 20: "Test all layers"] depict the systems architecture graphically with further explorations.

3.4.1 Import of .shp into DBMS

This part of the thesis explains the import of the spatial data from shapefile with extension .shp into two different Database Management Systems: PostgreSQL version 9.3 with spatial database extender PostGIS 2.1 and SQL Server Express in order to simulate similar dataflow as in SharePoint. For theoretical background about shapefile format see section [2.2.2].

There was given twelve shapefiles: four files for points, four files for lines and four files for polygons, which were imported into databases as separate project. The source for this project is attached this thesis, see Appendix.

Each file was assigned an unique Guid, called ListGuid in [Figure 9: Import into DBMS], and each feature had its own Guid, called ItemGuid, in order to simulate SPList, for more information see section [3.1].

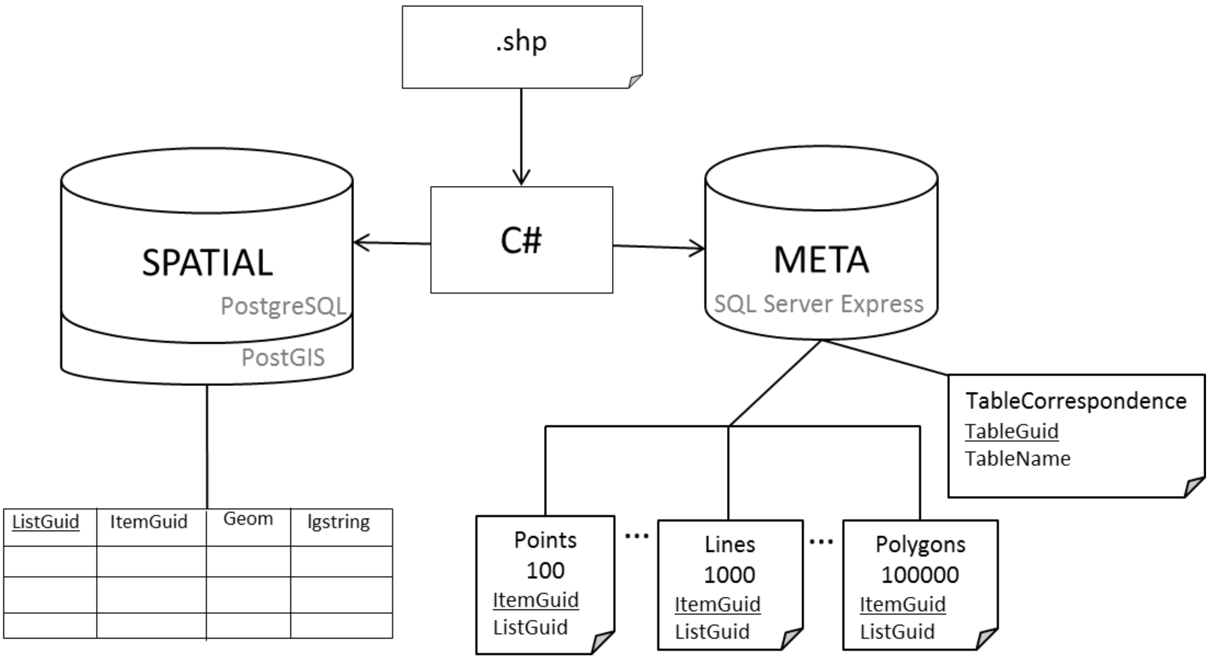


Figure 9: Import into DBMS

The reading of each file was implemented in three steps. During the first step the ListGuid for the file and the ItemGuid for each feature in the file were written into appropriate table in META (SQL Server Express). During the second step the ListGuid, called now TableGuid and the file name, called TableName in [Figure 9: Import into DBMS] were written into the table TableCorrespondence in META, see [Figure 10: TableCorrespondence in SQL Express].

	TableGuid	TableName
1	0f662f74-073f-410b-9d53-52db6b4307ff	lines_1k
2	2fb190d4-da8f-4527-b71d-8af5b1a6d9c4	lines_100k
3	3d0f5d84-3840-4ad5-ae00-c1e139d6f223	polygons_100k
4	4c36b07a-bb38-4e99-9947-875ab819f8c0	polygons_100
5	60a3b183-d278-43c5-bc60-dd5620a96cb6	points_100k
6	6bc8485a-f1f5-4bd2-bc61-aeff5c6dbaa4	lines_100
7	a2877fec-7b58-410a-a97a-ba51aa80963f	polygons_1k
8	b3ef7444-09e2-451c-9bec-1e50f19a3592	points_100
9	d4dd2d25-a3f5-457b-ba99-2c1b766b21db	points_1k
10	d89b7db4-e1ea-4b52-89b0-63d7d5e46c81	points_10k
11	f4b3acbb-d3d3-4394-af08-5688ae065157	polygons_10k
12	f54777f6-c43e-4928-8306-a89ae95c5d10	lines_10k

Figure 10: TableCorrespondence in SQL Express

During the third step ListGuid, ItemGuid and Geom were written into table SPATIAL (PostgreSQL) Table SPATIAL was done in two steps, in order to get possibilities to store spatial data correctly. For explanation about how PostGIS extension turns PostgreSQL into a spatial database see section [2.2.4].

The first step was to create a table in common way:

```
CREATE TABLE spatial(ListGuid UUID Not Null,ItemGuid Integer Not Null)
PRIMARY KEY (ListGuid, ItemGuid);
```

The second step was to add a geometry column:

```
SELECT AddGeometryColumn('spatial','geom',4326,'GEOMETRY',2);
```

Writing geom to this table was done by using `ST_GeomFromText(text WKT, integer srid)` function of PostGIS [21].

```
geom = "ST_GeomFromText('"+pGeom+"'),4326)"
sql = String.Format("INSERT INTO spatial(ListGuid,
ItemGuid,geom)VALUES({0},{1},{2}),ListGuid,ItemGuid,geom)";
```

In order to see how geom is stored in a database use function `ST_ASText(geom)`. Example of geom for point, line and polygon is shown in [Figure 11: Geom for a point, a line and a polygon].

```
                  geom
-----------------------------------------
 POINT(12.6225657231534 63.1098638917448)
 MULTILINESTRING((12.6216124070859 63.1093989151001,12.6215390603863 63.110295
61722,12.6235190696804 63.1103288620832,12.6235923554609 63.1094321197283,12.6
6124070859 63.1093989151001))
 POLYGON((12.6216124070859 63.1093989151001,12.6215390603863 63.1102956561722,
.6235190696804 63.1103288620832,12.6235923554609 63.1094321197283,12.621612407
59 63.1093989151001))
```

Figure 11: Geom for a point, a line and a polygon

To sum up, a separation of storing spatial data in a spatial database (PostgreSQL with PostGIS extension) and attribute data in a relational database (SQL Express) allows support for different combinations and includes a possibility for the components to evolve independently. The trade-off of this separation is the amount of components which build up the system.

3.4.2 GeoServer connection to PostgreSQL

This part of the thesis describes how to connect a GeoServer to a PostgreSQL database, how to publish a layer in the GeoServer with features that is configured against a table in the database and how to publish a SQL View that allows executing a custom SQL query with parameter supplied in the request to the layer. For more detailed information about creating and using a parametric SQL View see [23].

GeoServer should be connected to repositories where the spatial data is located by using Stores, a brief introduction to administrative interface of the GeoServer is found in section [2.2.3]. Each Store must be in the Workspace in order to use REST more effectively. When creating a new data store there are a few formats available classified in two types: Vector data sources and Raster data sources. The connection to PostgreSQL occurs by choosing PostGIS Database resource and saving access information such as username and password in the GeoServer, see [Figure 12: GeoServer connection to PostgreSQL].

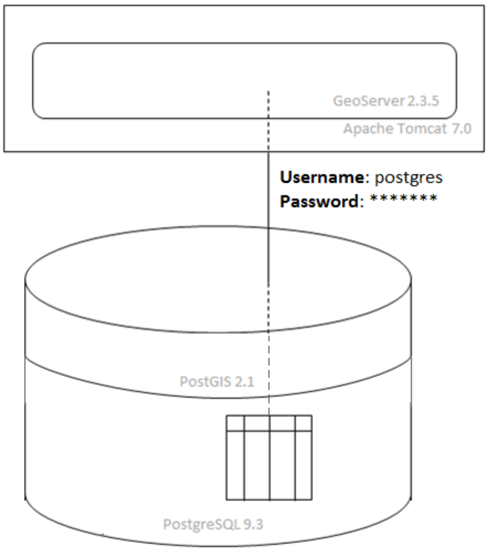


Figure 12: GeoServer connection to PostgreSQL

A new layer is added by choosing *Add a new resource* from the section Layers on left-side-hand of the administrative interface, see [Figure 2: GeoServer administrative interface] and published in order to save the configuration of the layer. The layer in GeoServer holds the metadata information about a feature such as the type of the layer, the Workspace and Store values for each layer, the name of the layer and if it is enabled for services such WMS, WFS and finally the Native SRS values, see [Figure 13: Layers published by GeoServer].

	Type	Workspace	Store	Layer Name	Enabled?	Native SRS
		MD	PostGreSQL	lines	✓	EPSG:4326
		MD	PostGreSQL	lines_limit	✓	EPSG:4326
		MD	PostGreSQL	points	✓	EPSG:4326
		MD	PostGreSQL	points_limit	✓	EPSG:4326
		MD	PostGreSQL	polygons	✓	EPSG:4326
		MD	PostGreSQL	polygons_limit	✓	EPSG:4326
		MD	PostGreSQL	spatial	✓	EPSG:4326

Figure 13: Layers published by GeoServer

The traditional way to access database data is to configure layers in GeoServer against either tables or database views. Starting with GeoServer 2.1.0, layers can also be defined as SQL view that allows send parameter to GeoServer using WMS or WFS requests [23]. A SQL View is created by choosing link *Configure New SQL View* from the *Add a new resource* on the Layer page. Within the SQL View query parameter names are delimited by leading and trailing % signs, see [Figure 14: SQL View].

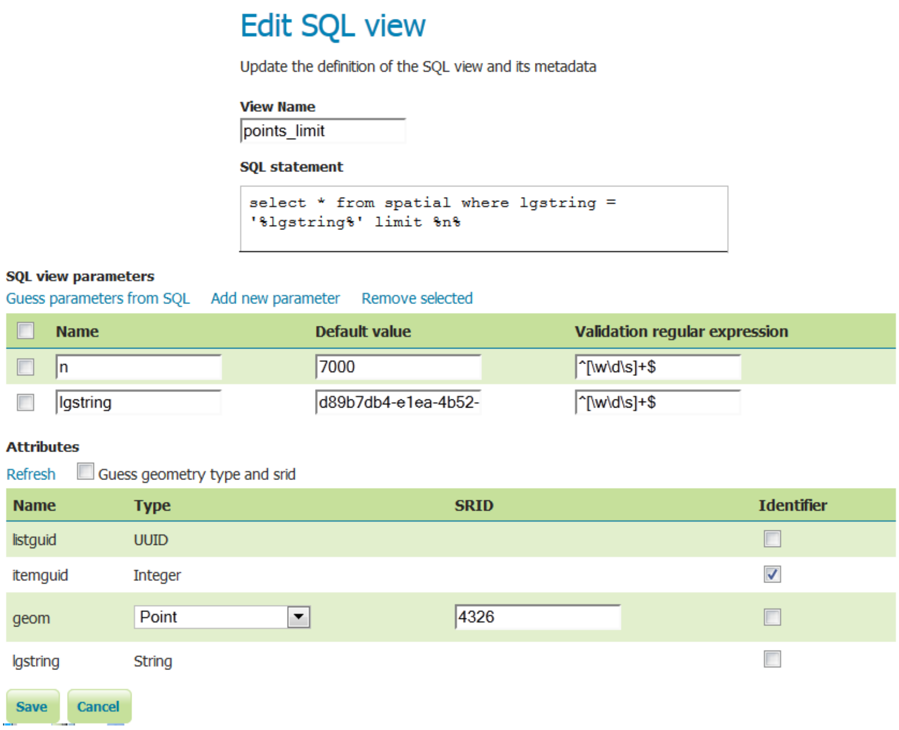


Figure 14: SQL View

Default values should be supplied for parameters and input values should be validated by Regular Expressions in order to eliminate risk of SQL injection attacks. The desired amount of features in the layer can be displayed by using CQL-filter, see a part of request using REST in Firebug [Figure 15: CQL Filter in Firebug].

Figure 15: CQL Filter in Firebug

The responce from GeoServer using cql_filter: lines_100 is shown in [Figure 16: CQL response from GeoServer in Firebug].

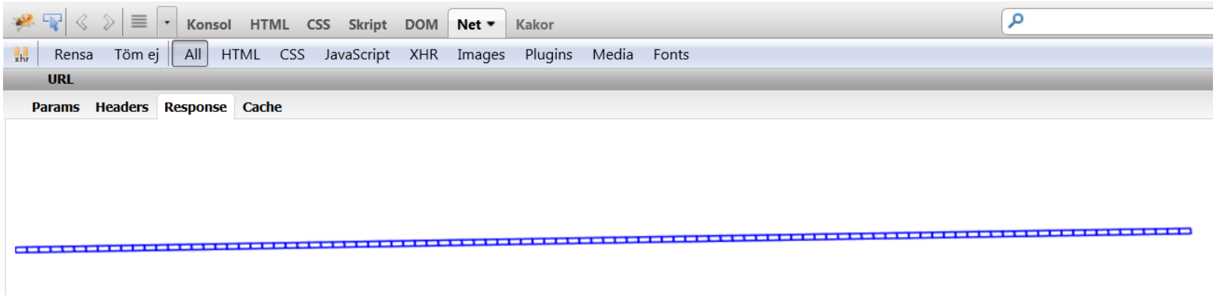


Figure 16: CQL response from GeoServer in Firebug

The attributes that can be used in the CQL filter are those included in the layer expressed by using Extended Common Query Language (ECQL). GeoServer supports a variety of vendor-specific WMS parameters.

3.4.3 GUI testbed

The figure [Figure 17 Default.aspx] shows the start page of the simplified Graphical User Interface testbed which was built by using XHTML and JavaScript. The rectangular area at the bottom of the testbed is a container for displaying a map with navigation and zoom possibilities displayed in the leftmost upper corner. The activating of the plus sign symbol in the rightmost upper corner allows possibility to see all added layers onto the map. The image size returned by GeoServer is very large (20206 x 330 px) and requires a large <div></div> container in order to be displayed completely. Because the complete visualization of a layer was not a primary goal of the project, the trade-offs including limiting the size of the <div></div> container and the size of the image to (2970x330) were done, that's why the default layer with 100 points (implementation of this layer shown in [Table 6 Implementation of the layer by using OpenLayers API]) is presented as a line in [Figure 17 Default.aspx].

exapmle:

```
var map = new OpenLayers.Map('map');
var wms_layer = new OpenLayers.Layer.WMS(
              "MD:points-Untiled",
              "http://localhost:8080/geoserver/MD/wms",
              {
                  LAYERS: “MD:points”,
                  STYLES: “”,
                  format: “image/png”,
                  CQL_FILTER: "lgstring='b3ef7444-09e2-451c-9bec-1e50f19a3592'"
             },
             {
                 isBaseLayer: true,
                 ratio: 1,
                 opacity: 0.5,
                 singleTile: true,
                 yx: { 'EPSG:4326': true }
             }
);
map.addLayer(wms_layer);
```

Table 6 Implementation of the layer by using OpenLayers API

This testbed includes three different types of test: “Test one layer”, “Test one zoom” and “Test all layers”. The first test “Test one layer” allows a user the possibility to test each appropriate layer separately by choosing the layer of the interest and pressing the button. In the real application a layer with points could mean a layer with some features such as towns in a country, or properties in a town, or amount of trees in the forest etc. A layer with lines could represent a continued in the space feature, such as roads, electric cables, borders, rivers etc. A layer with polygons could represent a feature with some area, such as countries, towns, pollution area etc.

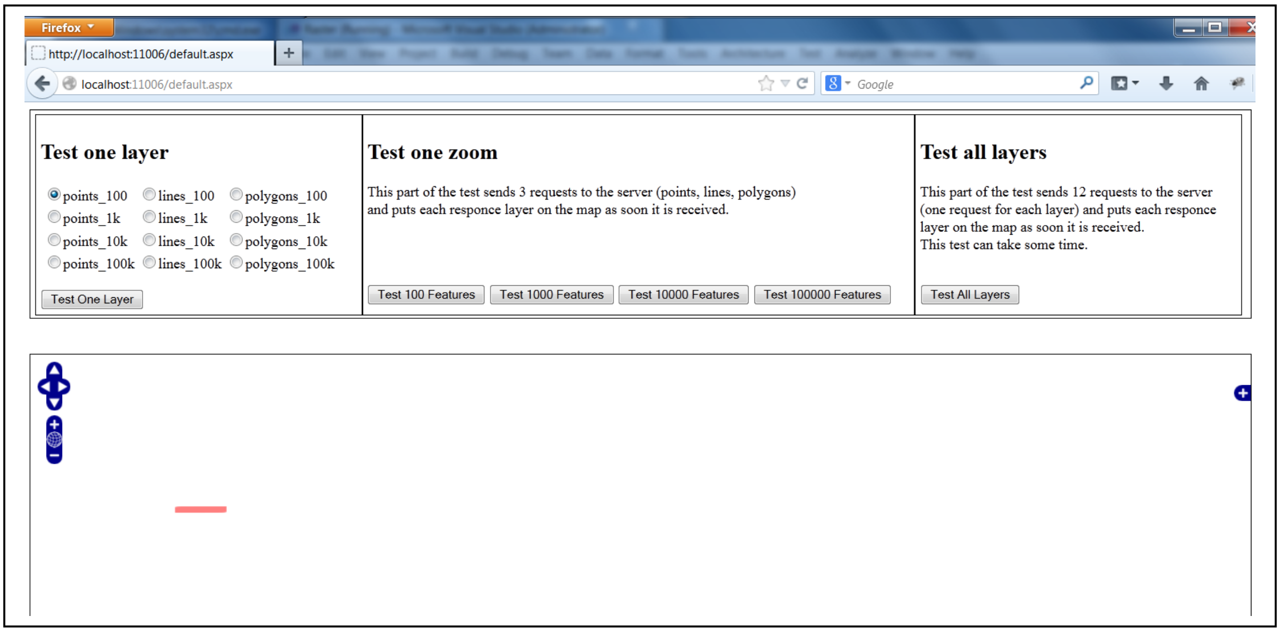


Figure 17 Default.aspx

The amount of features in the layer increases or decreases, according to the zoom, which imply in the background of the application a sending of the asynchronous request to the map server to get a layer with appropriate number of features. The result of this test is presented in section [4.1.1] and shown in [Figure 18: One layer test for 100 000 points].

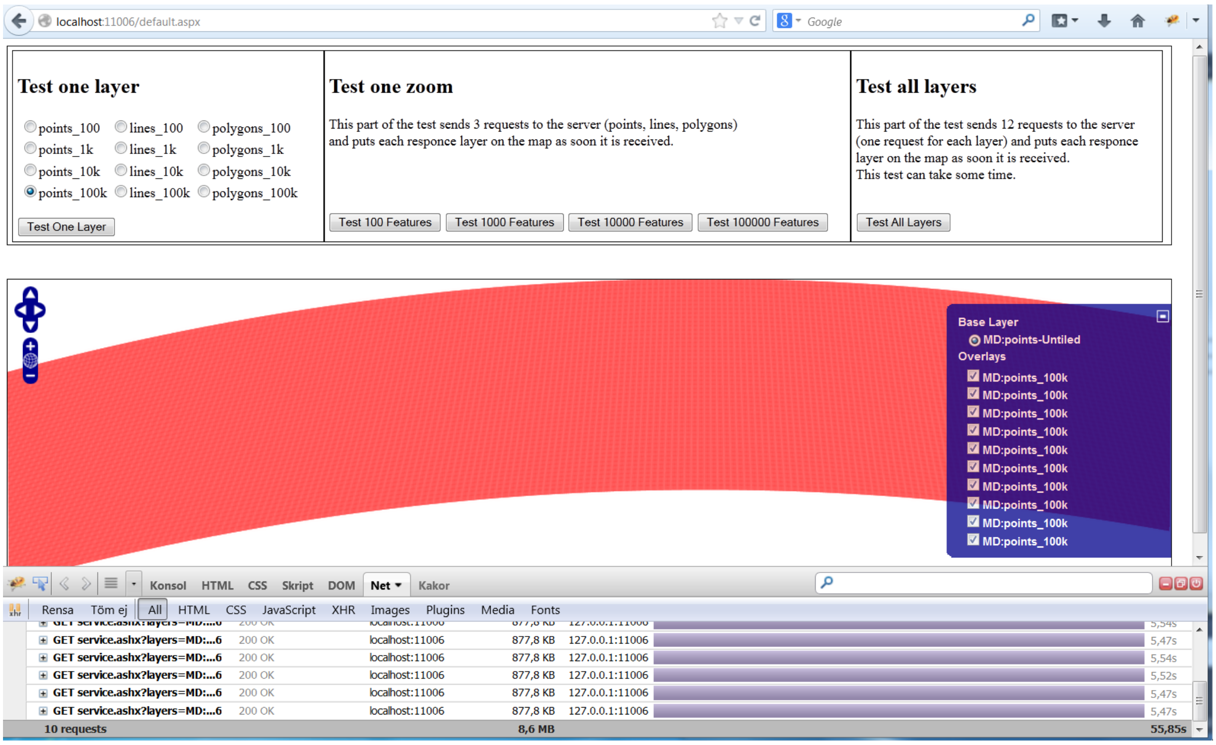


Figure 18: One layer test for 100 000 points

The second test, called "Test one zoom" allows a user to test a map with three different layers including the same amount of features. For example, by pressing button "Test 100 Features" the application sends three requests to the map server, one to get a layer with appropriate number of points, one to get a layer with the same number of lines, and one to get a layer with the same number of polygons. The result of this test is described in section [4.1.3] and the request using Firebug for three layers with the same amount of features is displayed in the bottom of [Figure 19: Request for zoom with 100 features].

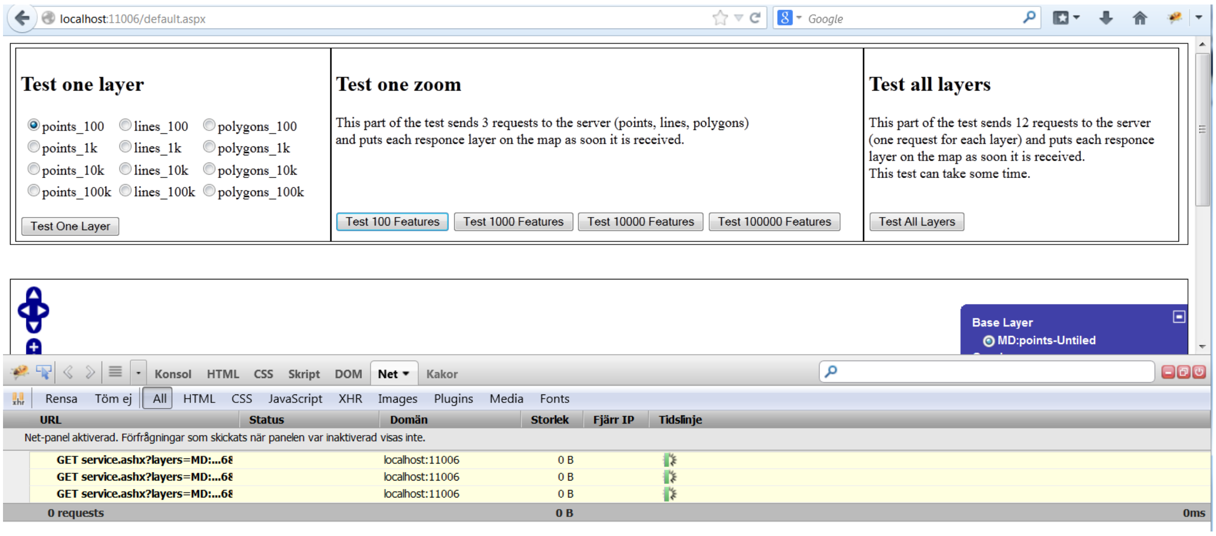


Figure 19: Request for zoom with 100 features

The third test, called “Test all layers” allows a user to test a map with different layers including different amount of features. Hypothetically this could be a zoomed-in map that includes twelve layers with feature information according to the zoom level. For example, at zoom level number three there is a layer with a hundred of big polygons, say countries, a layer with a thousand of smaller polygons, say towns and villages, a layer with a ten thousand roads between towns, a layer with a hundred thousand properties etcetera. The result of this test is described in section [4.1.4] and response images layered on the map is shown in [Figure 20: “Test all layers”].

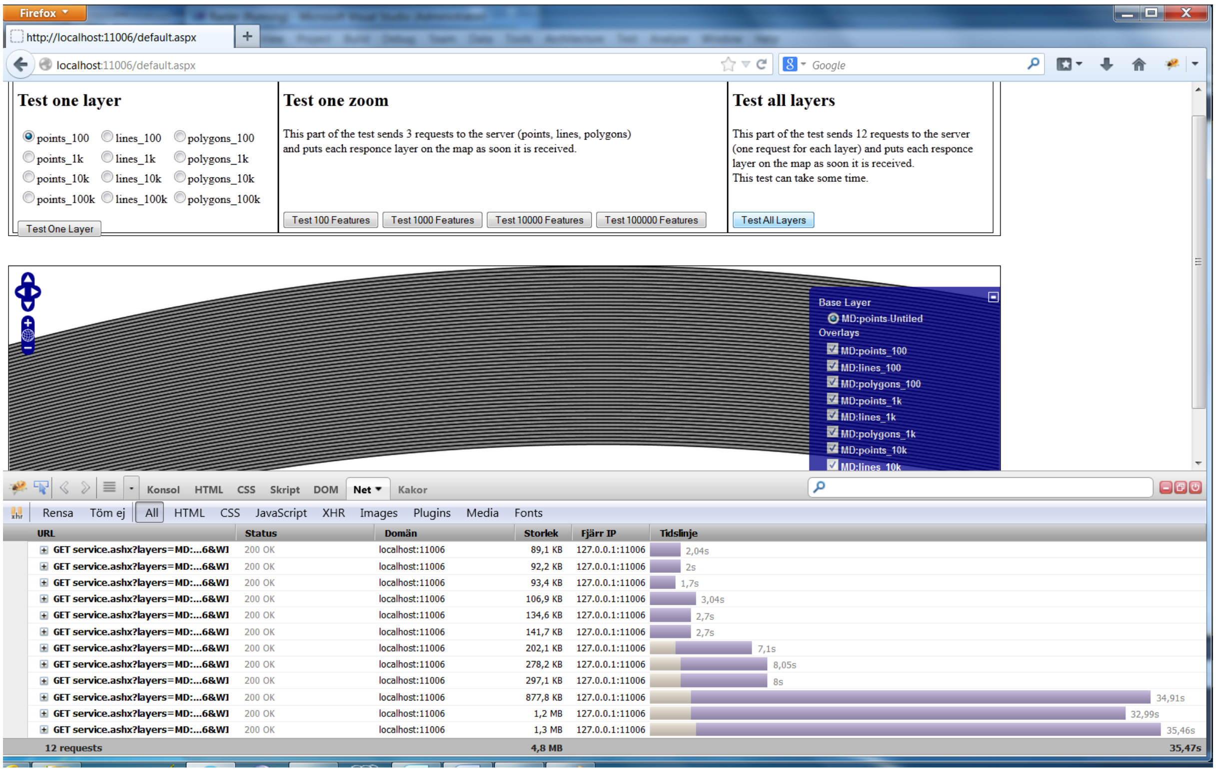


Figure 20: “Test all layers”

The Firebug part on the bottom of [Figure 20: “Test all layers”] shows time bars for each request a browser has to wait in order to get response with the image from the server. More about time needed to produce a request presented in chapter [4].

4 Test and result

This part of the thesis presents different types of tests with explanations. Tests presented in the section [4.1] were made in order to investigate the data flow in the implemented system. Test presented in section [4.2] is of an analytical nature and was made in order to study data flow between components in the system.

4.1 Tipping points

Tests presented during sections [4.1.1] to [4.1.4] are of an investigative nature and were made the same day. The result in all tests for each number of features was calculated as a mean value from twenty requests. It is worth to point out that the meaning with each test was to send only ten requests but each request was sent by the browser twice because of the size of the image, according OpenLayers API.

4.1.1 Test one layer

This test presents the mean time a user has to wait in order to get the chosen layer with the appropriate number of features. The idea behind this test is described in section [3.4.3]. The figures from [Figure 21: Overview diagram of "Test one layer" for points] to [Figure 23: Overview diagram of "Test one layer" for polygons] and the tables from [Table 7: Overview table of "Test one layer" for points] to [Table 9: Overview table of "Test one layer" for polygons] analyze the test result for each type of feature separately. The table [Table 10: "Test one layer": overview table] and the figure [Figure 24: "Test one layer": overview diagram] summarize the result in order to get a better understanding of data flow in the implemented system.

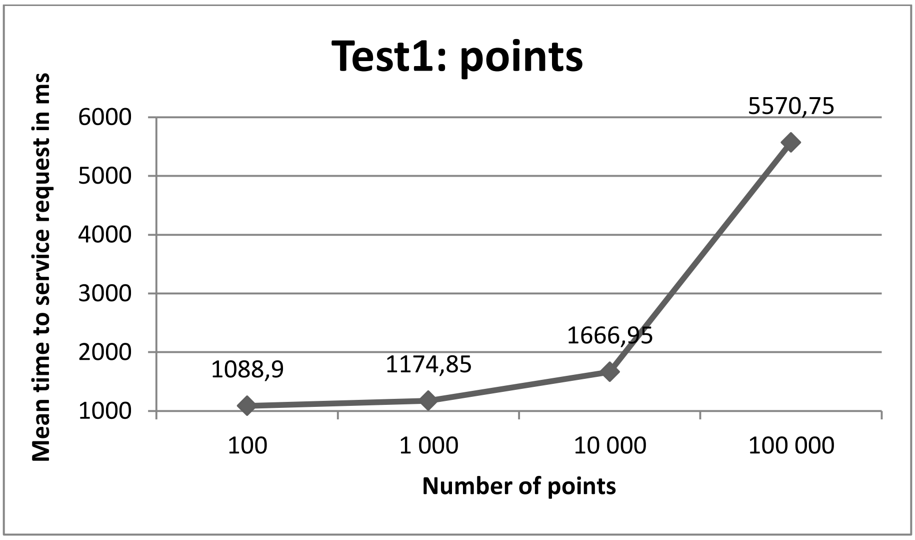


Figure 21: Overview diagram of "Test one layer" for points

The mean values of twenty requests for the hundred, the thousand, the ten thousand and the one hundred thousand points are presented with help of labels in [Figure 21: Overview diagram of "Test one layer" for points]. The percentage growth of time in milliseconds and the percentage growth of image size in KB are shown in [Table 7: Overview table of "Test one layer" for points].

Number of points	Mean time ms	% growth in ms from 100 points	Size of the image	% growth in KB from 100 points
100	1088,9	0%	89,1 KB	0%
1 000	1174,85	7,89 % (8%)	106,9 KB	19,98% (20%)
10 000	1666,95	53,09% (53%)	202,1 KB	126,82% (127%)
100 000	5570,75	411,59% (412%)	877,8 KB	885,19% (885%)

Table 7: Overview table of "Test one layer" for points

The time needed for rasterization and the size of the raster images increase very rapidly when number of requested points passes one thousand.

The similar tests with mean values of twenty requests for the hundred, the thousand, the ten thousand and the one hundred thousand lines are presented with help of labels in [Figure 22: Overview diagram of "Test one layer" for lines].

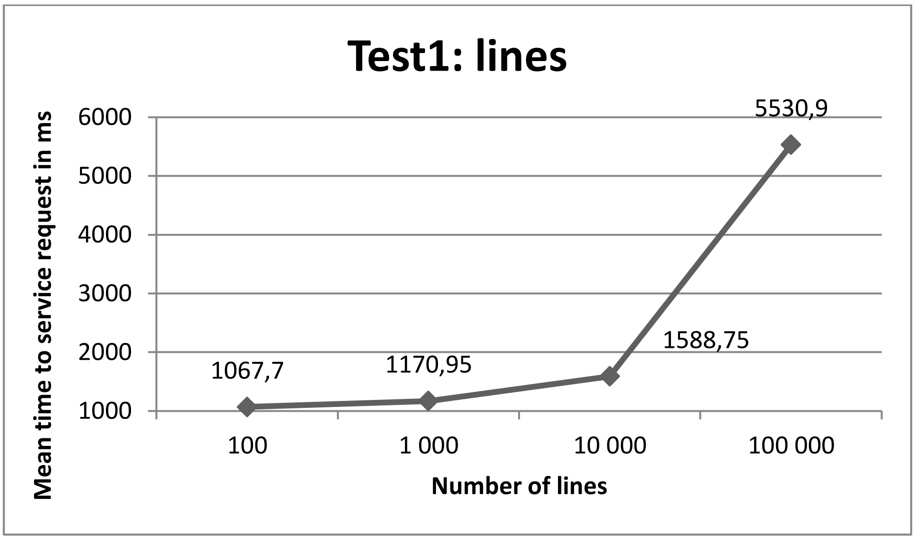


Figure 22: Overview diagram of “Test one layer” for lines

The results of percentage growth of time in milliseconds and the percentage growth of image size in KB are shown in [Table 8: Overview table of “Test one layer” for lines].

Number of lines	Mean time ms	% growth in ms from 100 lines	Size of the image	% growth in KB from 100 lines
100	1067,7	0%	92,2 KB	0%
1 000	1170,95	9,67% (10%)	134,6 KB	45,99% (46%)
10 000	1588,75	48,80% (49%)	278,2 KB	201,74% (202%)
100 000	5530,9	418,02% (418%)	1,2 MB 1228.8KB	1232.75% (1233%)

Table 8: Overview table of “Test one layer” for lines

Both figure [Figure 22: Overview diagram of “Test one layer” for lines] and table [Table 8: Overview table of “Test one layer” for lines] demonstrate the similar behavior for layers with lines as for layers with points, such as the time needed for rasterization and the size of the raster image increases rapidly when number of requested lines passes one thousand.

The test for polygons with mean values of twenty requests for the hundred, the thousand, the ten thousand and the one hundred thousand polygons are presented with help of labels in [Figure 23: Overview diagram of “Test one layer” for polygons]

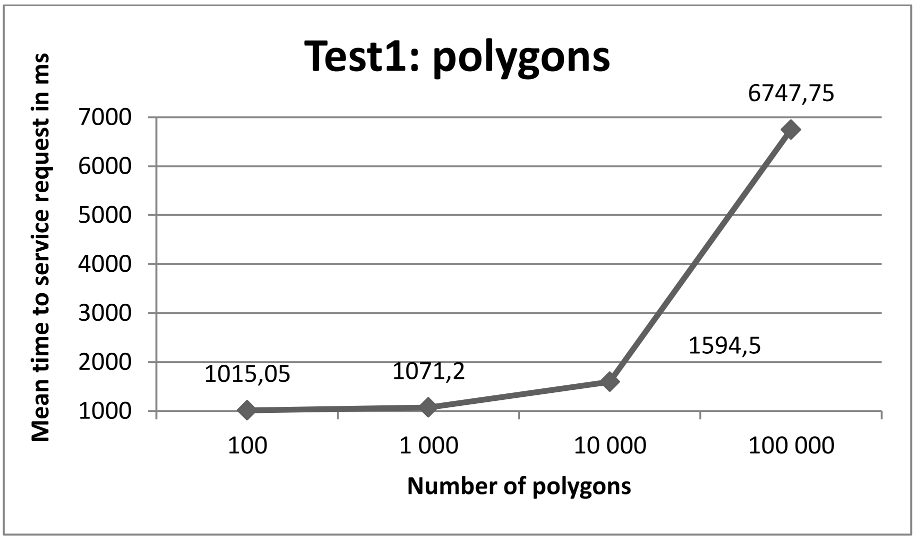


Figure 23: Overview diagram of "Test one layer" for polygons

The results of percentage growth of time in milliseconds and the percentage growth of image size in KB are shown in [Table 9: Overview table of "Test one layer" for polygons].

Number of: polygons	Mean time ms	% growth in ms from 100 polygons	Size of the image	% growth in KB from 100 polygons
100	1015,05	0%	93,4 KB	0%
1 000	1071,2	5,53% (6%)	147,7 KB	58,14% (58%)
10 000	1594,5	57,09% (57%)	297,2 KB	218,20% (218%)
100 000	6747,75	564,77% (565%)	1,3 MB 1331,2 KB	1325,27% (1325%)

Table 9: Overview table of "Test one layer" for polygons

The behavior of layers with polygons is similar to the behavior of layers with points and lines, such as the time needed for rasterization and the size of the raster image increase rapidly when number of requested polygons passes one thousand.

The overview of percentage growth for points, lines and polygons is summarized in [Table 10: "Test one layer": overview table].

TEST ONE LAYER: OVERVIEW

% growth in ms from 100 features			
Number of features	Points	Lines	Polygons
100	0%	0%	0%
1 000	8%	10%	6%
10 000	53%	49%	57%
100 000	412%	418%	565%

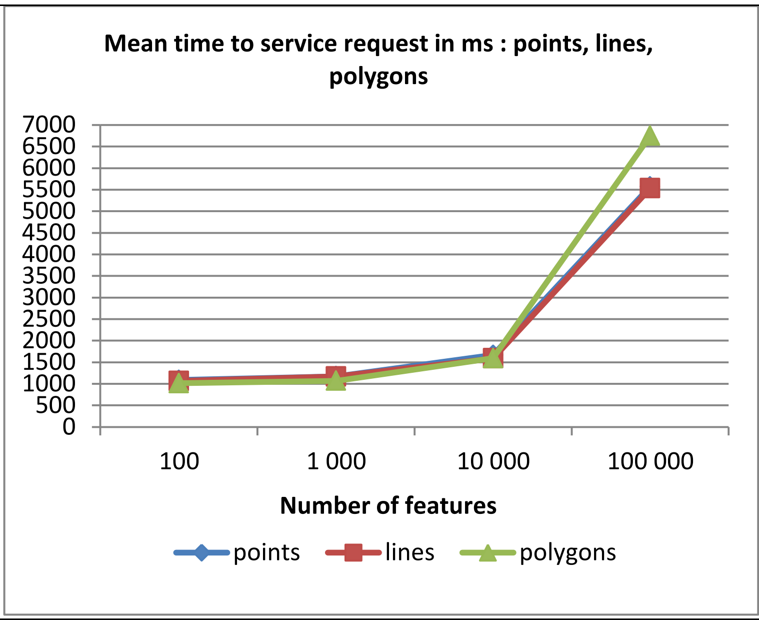


Table 10: "Test one layer": overview table

The summarizing of percentage growth for each type of feature separately demonstrates an interesting behavior of the time the server (Service.ashx) needed in order to generate the response image to the client. The comparison of the layers with one thousand features illustrates that Service.ashx generates the layer with polygons faster than the layer with points, or the layer with lines, regardless of the fact that the layer with polygons is bigger than each of the two other layers. The comparison of the layers with ten thousand features illustrates that the layer with lines is generated faster than each of the two other layers.

The summarizing of the mean time the Server.ashx needed in order to generate the response image illustrates that it takes more time to get the layer with points than the layer with lines or the layer with polygons, see [Figure 24: "Test one layer": overview diagram]. This result is relevant for layers with one hundred, one thousand and ten thousand features. The comparison of layers with lines and polygons for both one hundred and one thousand features illustrates similar results, that it takes more time to process the smaller image.

One possible explanation for that type of result can be the algorithm that GeoServer uses for converting spatial data to a raster. The second explanation can be the different performance in fetching data from database.

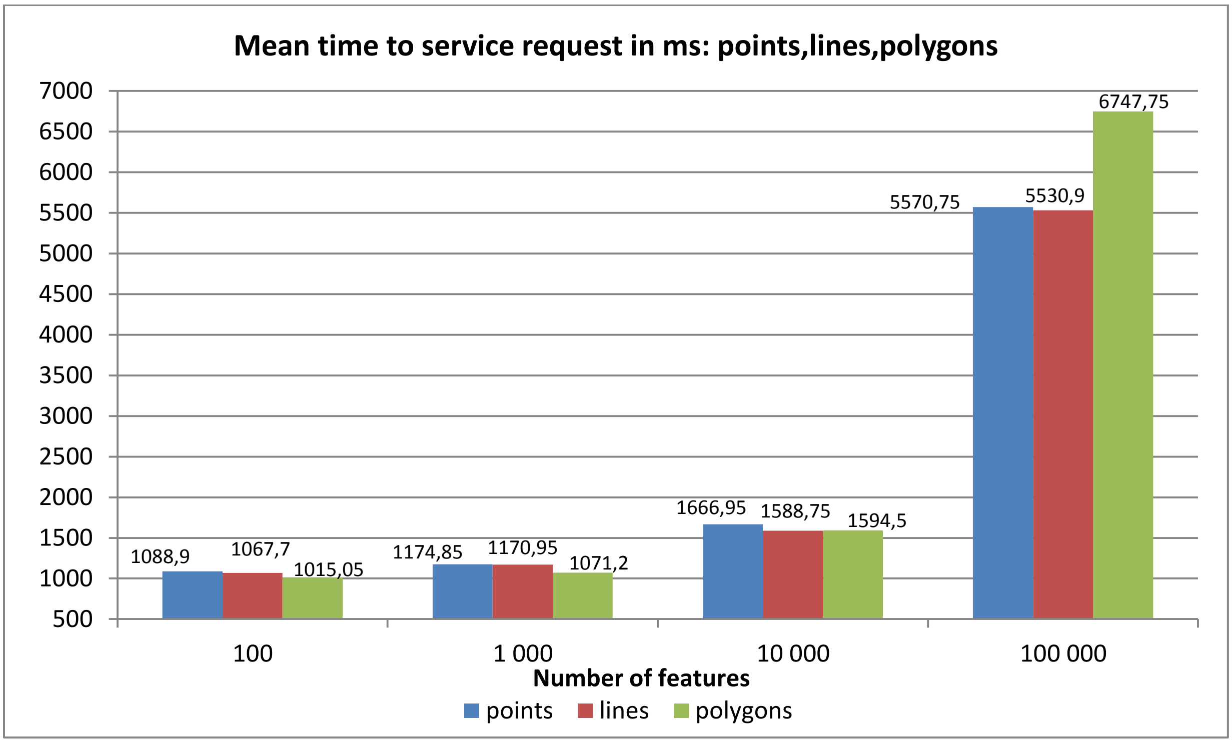


Figure 24: "Test one layer": overview diagram

The diagram in [Figure 24: "Test one layer": overview diagram] demonstrates the mean time the server Service.ashx needed to produce an image with requested number of features. The time the server needs to generate response with an image increases after one thousand features and growths rapidly after ten thousand features. This fact was investigated by using "Slice Test" and the result of this test is presented in section [4.1.2].

4.1.2 Slice test

This test investigates the time the server simulated by Service.ashx needs to generate an image with reduced number of features. All results from "Test one layer" indicated that it takes more time for the server to generate an image with points when a client asks for layers with one hundred, one thousand and ten thousand features. When the server gets a request for the layer with a hundred thousand features, the layer with polygons takes longer time than any of the other layers. This gave the idea to use "Slice test" using layers with the points up to ten thousand and after that use the layer with polygons.

In order to accomplish "Slice test" three additional layers: points_limit, lines_limit and polygons_limit were published in the GeoServer by using SQL View and cql-filter, see section [3.4.2].

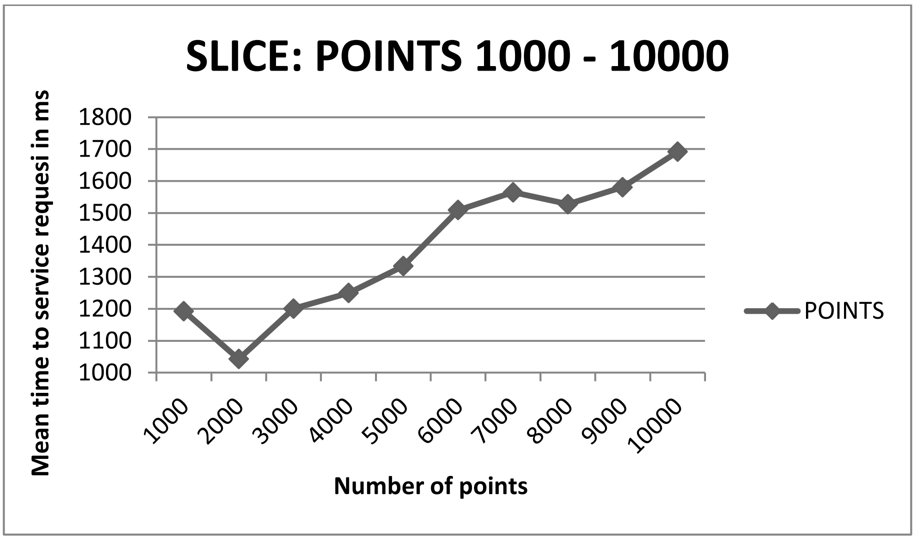


Figure 25: “Slice test” from 1000 to 10 000 points

The curve indicating mean time to service request in [Figure 25: “Slice test” from 1000 to 10 000 points] has almost linear growth except the result for 2000, 6000 and 7000 points. A possible explanation to this behavior can be the Garbage collection in C# (Visual Studio) or in Java (GeoServer) or management of the disk cache.

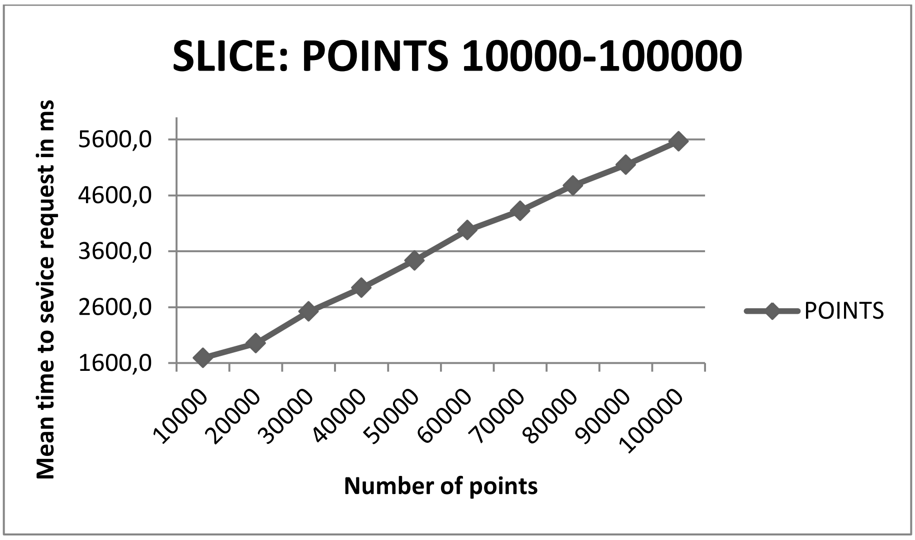


Figure 26: “Slice test” for 10000 to 100000 points

The curve for lines in the [Figure 26: “Slice test” for 10000 to 100000 points] and also the curve for polygons in [Figure 27: “Slice test” for 10000 to 100000 polygons] increases linearly.

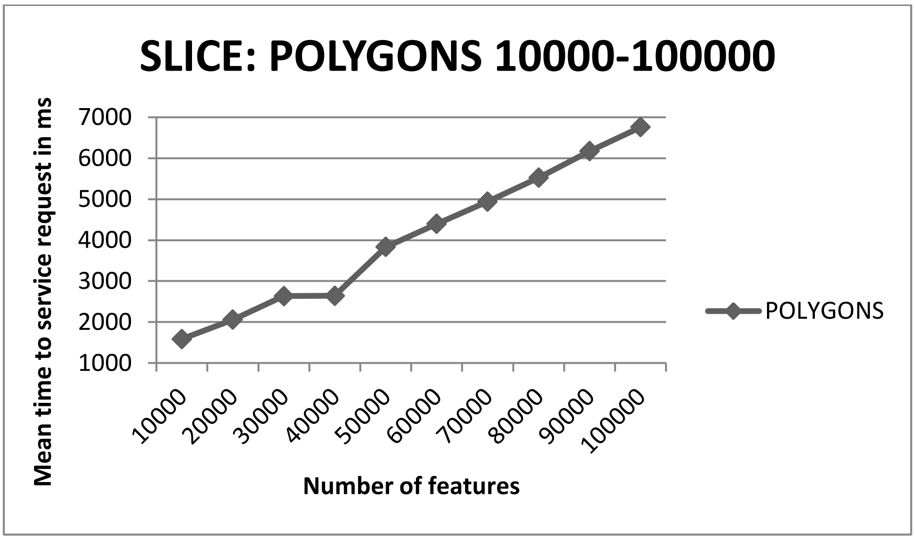


Figure 27: "Slice test" for 10000 to 100000 polygons

The more detailed investigation of the time that each part of the implemented system needed to process a request is presented in section [4.2].

4.1.3 Test one zoom

This test summarizes the maximum time a user has to wait to get a map with three layers including the same amount of features in the layer. The idea behind this test is presented in section [3.4.3]. Each zoom was tested separately and the maximum time for four "Zoom tests" is presented with help of labels in the [Figure 28: Overview diagram of "Test one zoom"]. The reason of choice to measure the maximum time in this test can be explained by the fact that the map considered being complete when all features which belong to the same zoom are displayed on the map.

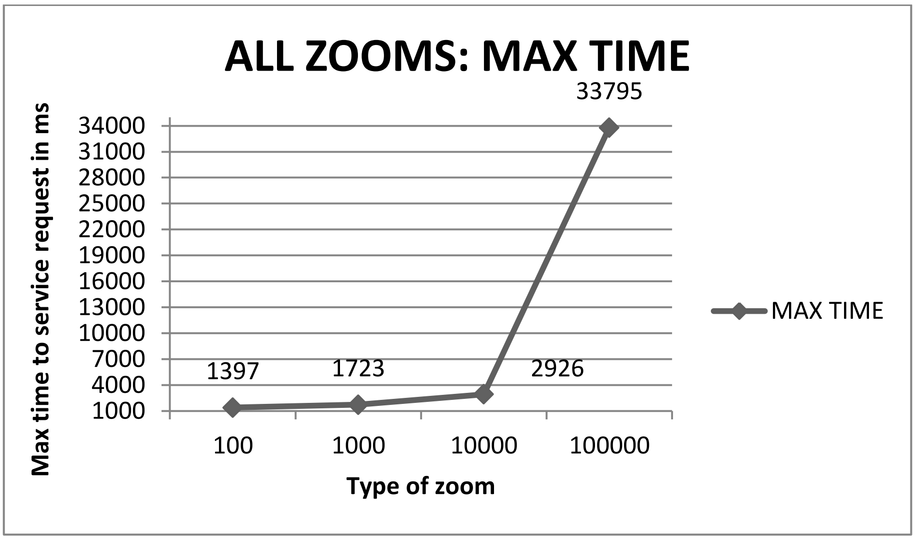


Figure 28: Overview diagram of “Test one zoom”

The curve in the [Figure 28: Overview diagram of “Test one zoom”] growths also rapidly as the curves for points, lines and polygons in section [Test one layer] but with a little bigger percentage increase of 23,34% between zoom 100 and zoom 1000, and 109,45% between zoom 100 and zoom 10000, and 2319,11% between zoom 100 and zoom 100000.

The break point for the curve acceleration in this diagram also starts after the layer with 10000 features. This observation illustrates that the implemented system can generate layers with up to 10000 features relatively fast. After this breakpoint the system works much slower.

4.1.4 Test all layers

This test also measure the maximum time a user has to wait in order to get a map with all twelve layers including different amount of features in each layer, with a similar explanation as in section [4.1.3] that the map considered being complete when all features which have to be displayed are on the map. The idea behind this test is presented in section [3.4.3].

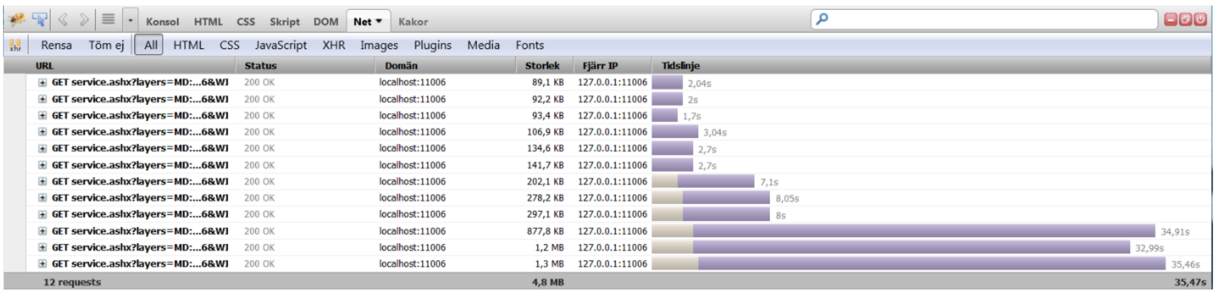


Figure 29: “Test all layers” test in Firebug

The twelve time bars in Firebug shown in the [Figure 29: "Test all layers" test in Firebug] illustrate the time the browser waits for each layer. As it was observed in the section [4.1.1] and [4.1.3] the layers with one hundred and one thousand features are generated relatively fast and after that the time to wait increases rapidly.

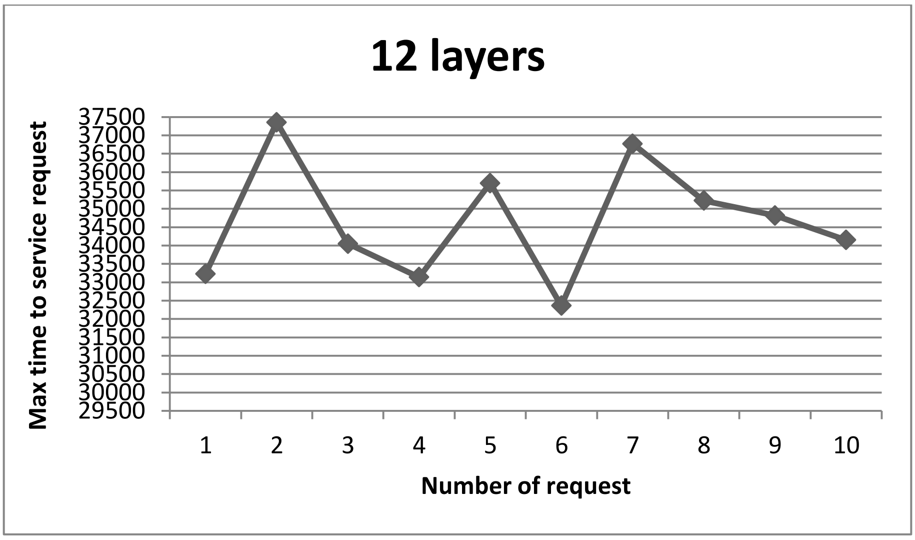


Figure 30: Overview diagram of "Test all layers"

This test was repeated ten times and the maximum time for each request is shown in the [Figure 30: Overview diagram of "Test all layers"]. The range of the time is between [32000, 37500] milliseconds. A possible explanation to this behavior can be the Garbage collection in C# (Visual Studio) or in Java (GeoServer) or management of the disk cache.

4.2 Bottleneck

This test analyzes the maximum time the components of the implemented system, see [Figure 8 System architecture] need in order to generate the response image. The test was made two days later, but the diagram in the figure [Figure 31: Service.ashx service time for a request] is much similar to the results illustrated in the diagram [Figure 24: "Test one layer": overview diagram] which confirms reliability of the tests presented in section [4.1.1].

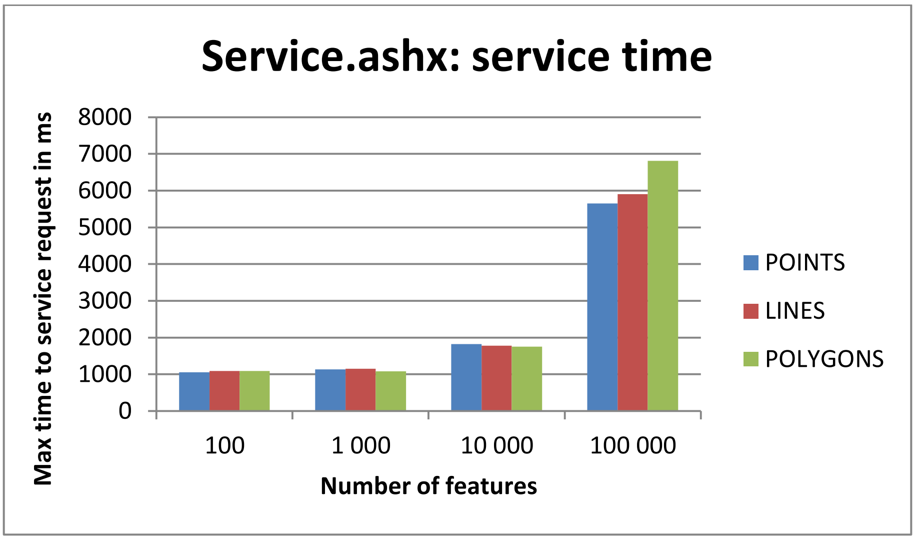


Figure 31: Service.ashx service time for a request

The time presented in the diagram in [Figure 31: Service.ashx service time for a request] is the time between client module and the C# module in the [Figure 8 System architecture] that symbolizes the server Service.ashx. The next diagram summarize the time needed to ask SQL Express database called META in the figure about the Guid of the table with requested number of features and generate the answer.

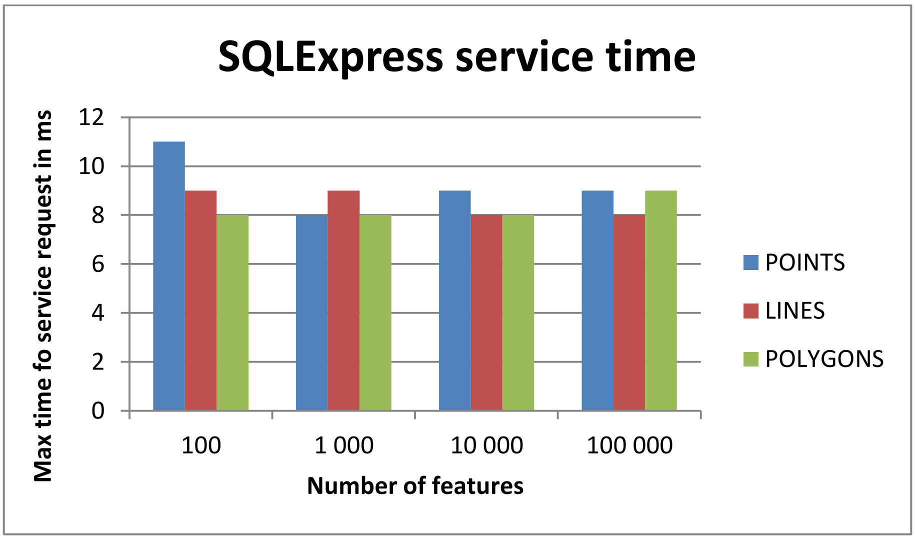


Figure 32: SQL Express service time for a request

The maximum time for connection and serving SQL query oscillates between eight and eleven milliseconds [Figure 32: SQL Express service time for a request]. In this project SQL query was limited to index-lookup, which is usually fast and should go in constant time. The possible explanation to this oscillation is that there was no data in disk cache.

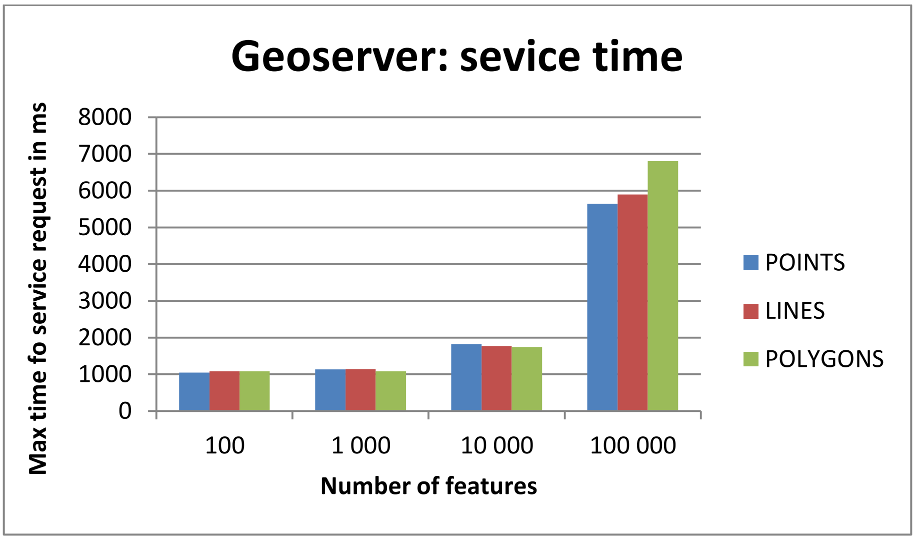


Figure 33: GeoServer: service time for a request

The next diagram [Figure 33: GeoServer: service time for a request] summarizes the time the GeoServer needed to service a request and that diagram defines the critical module in the [Figure 8 System architecture].

To sum up, there is one apparent critical module in the system that delays data flow in the system when the number of requested features passes one thousand and it can slow down data flow when the number of requested features passes ten thousand.

5 Problem and solution

This part of the thesis summarizes the problems that encountered during interrogation of the components of the system and discussed the limitations of the system occurred during the tests. A schematic overview of all occurred problems is presented in [Figure 34: Problems occurred in the project].

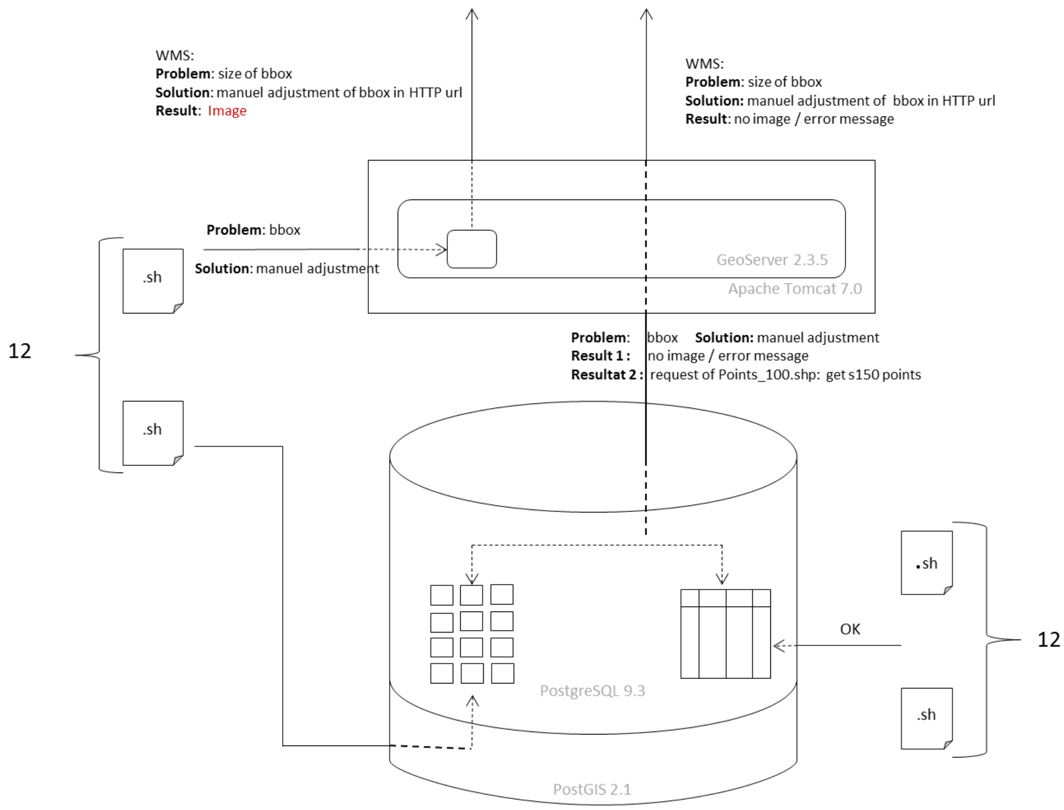


Figure 34: Problems occurred in the project

The first difficulty occurred while loading data into PostGIS and publishing a layer with features in the GeoServer, the rightmost part of the [Figure 34: Problems occurred in the project]. This problem had its origin in the way coordinates of native Coordinate Reference System (SRS) was stored in the GeoServer, for more information about SRS and SRID see [2.2.1]. Shape files that were used in the project were generated using ArcGIS SWEREF99_TM projection. SRID that represents that projection is presented in the rightmost column of the [Table 11: SWEREF99_TM Bounding Boxes problem].

Coordinates of that projection were deformed after the publishing of the layer (explained using print screen images in the leftmost column of the table [Table 11: SWEREF99_TM

Bounding Boxes problem], which resulted in the problem to compute correct Bounding Box Area.

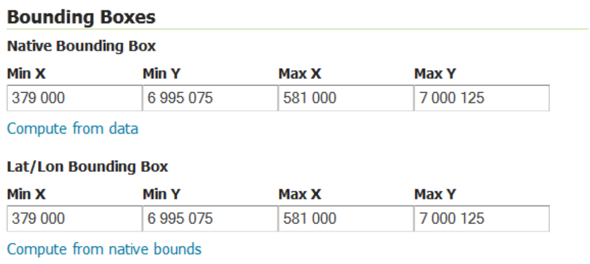
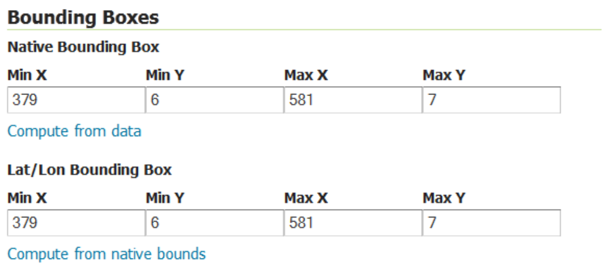
<table>
<tr><th>SWEREF99_TM before saving</th><th>SWEREF99_TM</th></tr>
<tr><td>
</td>
<td rowspan="3"><pre>PROJCS["SWEREF99 TM",
  GEOGCS["SWEREF99",
    DATUM["SWEREF99",
      SPHEROID["GRS 1980", 6378137.0,
       298.257222101,
      AUTHORITY["EPSG","7019"]],
      TOWGS84[0.0, 0.0, 0.0, 0.0, 0.0,0.0,
              0.0],
      AUTHORITY["EPSG","6619"]],
    PRIMEM["Greenwich", 0.0,
AUTHORITY["EPSG","8901"]],
    UNIT["degree", 0.017453292519943295],
    AXIS["Geodetic longitude", EAST],
    AXIS["Geodetic latitude", NORTH],
    AUTHORITY["EPSG","4619"]],
  PROJECTION["Transverse_Mercator",
AUTHORITY["EPSG","9807"]],
  PARAMETER["central_meridian", 15.0],
  PARAMETER["latitude_of_origin", 0.0],
  PARAMETER["scale_factor", 0.9996],
  PARAMETER["false_easting", 500000.0],
  PARAMETER["false_northing", 0.0],
  UNIT["m", 1.0],
  AXIS["Easting", EAST],
  AXIS["Northing", NORTH],
  AUTHORITY["EPSG","3006"]]</pre></td></tr>
<tr><th>SWEREF99_TM after saving</th></tr>
<tr><td>
</td></tr>
</table>

Table 11: SWEREF99_TM Bounding Boxes problem

It was done two different approaches, which are schematically presented in the leftmost part of the [Figure 34: Problems occurred in the project] in order to solve the problem.

The first approach was about the way the shape files were loaded into PostgreSQL. During this approach the spatial data stored in the shape files were loaded using PostGIS Shapefile Import/Export Manager, which resulted in generating twelve tables, one table for each file. This approach did not eliminate Bounding Box problem mentioned earlier.

During the second approach the spatial data from the shape files were loaded directly into the GeoServer. This approach did not eliminate Bounding Box problem, however it helped to understand the problem, because the manual adjustment of bbox parameters in the HTTP url (underlined in the [Table 12: Manual adjustment of bbox parameters in HTTP request] generated a wanted image.

HTTP request:
http://localhost:8080/geoserver/MD/wms?service=WMS&version=1.1.0&request=GetMap&layers=MD:points_100&styles=&**bbox=380000.0,7000049.0,382000.0,7000051.0**&width=2970&height=330&srs=EPSG:4326&format=image%2Fpng

Table 12: Manual adjustment of bbox parameters in HTTP request

The problem was solved by contacting Sweco GeoServer expert and updating GeomSRID in spatial table in PostgreSQL database from SWEREF99_TM projection into WGS 84 projection. SRID of this projection and stored coordinates are shown in [Table 13: WGS 84 projection].

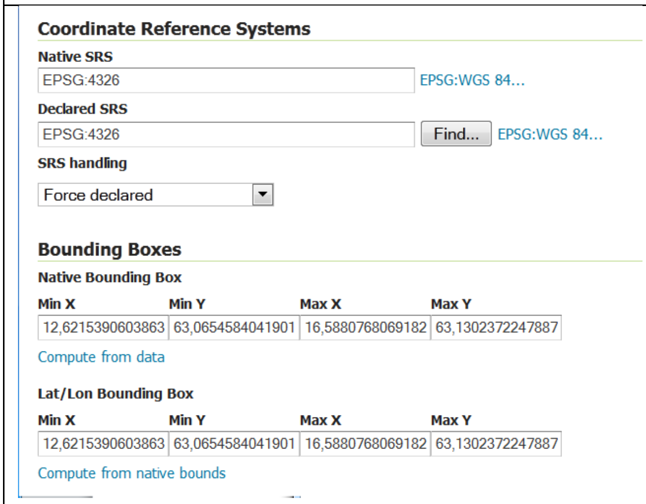
<table>
<tr><th>WGS 84 (EPSG:4326)</th><th>WGS 84 (EPSG:4326)</th></tr>
<tr><td>
</td><td>GEOGCS["WGS 84",
DATUM["World Geodetic System 1984",
SPHEROID["WGS 84", 6378137.0,
298.257223563,
AUTHORITY["EPSG","7030"]],
AUTHORITY["EPSG","6326"]],
PRIMEM["Greenwich", 0.0,
AUTHORITY["EPSG","8901"]],
UNIT["degree", 0.017453292519943295],
AXIS["Geodetic longitude", EAST],
AXIS["Geodetic latitude", NORTH],
AUTHORITY["EPSG","4326"]]</td></tr>
</table>

Table 13: WGS 84 projection

To sum up, the determining of the Bounding Box problem, the understanding of how this problem affects the possibility to use WMS service, and finally the generating of a correct solution – all these obstacles delayed the project in several days.

There are three limitations in the system that could not be solved completely because of limit of time. The first limitation appeared during the "Test one layer" test while observing tests in Firebug. The delay between requests was set to five seconds and after a while GeoServer could not generate the requested image, see [Figure 35: Error occurred during "Test one layer" test in Firebug]. The test was stopped after that observation and delay was changed to 30 seconds and during the test "Test all layers" the delay was set to one minute.

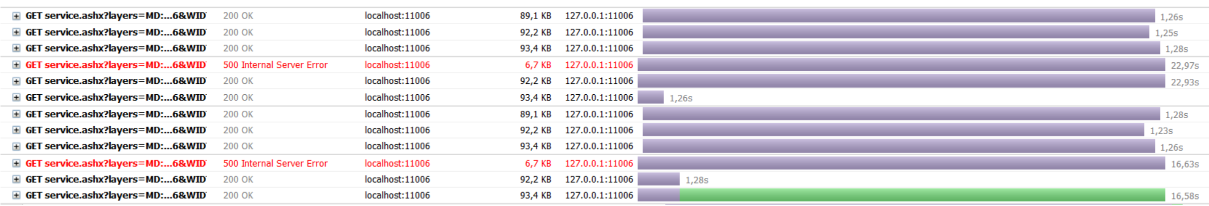


Figure 35: Error occurred during "Test one layer" test in Firebug

The second limitation occurred when analyzing the TimeLog after the first test was run completely. Probably because of maintaining disk cache or because of some misunderstanding of built-in functionality in Visual Studio and how ASP.NET works, the print-out was delayed and appeared much later. The quick solution to this limitation was manual control of the content in the TimeLog after each request, which was possible because of the changed delay between requests.

The third limitation is the direct consequence of the image size according to OpenLayers API. When the image map is big enough, the browser sends a second request to the server directly after the first request. Any solution to this problem was not found. Because of that fact, the mean value was calculated for twenty requests, which was mentioned earlier in chapter [4].

6 Conclusion and future work

The goal of the project presented in this thesis was to investigate the possibilities to the rasterization of the fragmented spatial data between GIS and Microsoft Share Point platform. This project was presented by Sweco Position that have built the system SMIL which offers the possibility to complement the information stored in the organization with spatial support. This organization was interested in analyzing the functionality of the built system.

The purpose of the project was to measure and analyze the time during rasterization of the fragmented spatial data by implementing a much more simplified version with similar organization of data storage and data flow as the existing system. Because of the fact that the only functionality of the Share Point platform and not the Share Point itself was of interest in the project, the trade-off about implementation resulted in the agreement that the implemented version should simulate the demanded functionality of Share Point without using this software in the project.

The performance should be measured for one hundred, one thousand, ten thousand and one hundred thousand features, such as points, lines and polygons. The first milestone was to identify any tipping points during rasterization in the implemented system. This milestone was achieved by applying four different types of tests: "Test one layer", "Slice test", "Test one zoom" and "Test all layers", which illustrated the similar result that the time needed for rasterization and the size of the raster image increases rapidly when the number of requested features passes one thousand. The second observation during these tests was identified by comparison of layers with different types of features. This comparison illustrates that the server simulated by Service.ashx needed more time in order to generate the response layer with points than the layer with lines or the layer with polygons, which are bigger in size. This result is relevant for layers with one hundred, one thousand and ten thousand features. The possible explanation for that type of result could be the algorithm that GeoServer uses for converting spatial data to a raster. The analysis of that algorithm and any improvement of the performance could be a possible specialty proposal for future investigation.

The second milestone was to identify any possible bottlenecks between modules involved in the system while generating response, if the occurrence of tipping points was observed. The detailed observation identified the GeoServer as an apparent critical module in the system that delays data flow when the number of requested features passes one thousand and it can slow down the system when the number of requested features passes ten thousand. It could be of interest to investigate the possibility of splitting requests for the layers with large number of features in order to improve the service time the GeoServer needs for working with a request.

This project was a very interesting investigation with a lot of challenges and experiences about how open source software components interoperate with software systems developed by Microsoft.

7 References

1. Cuthbert, A., *User Interaction with Geospatial Data*, OGC Document #98-060, October 1998

2. Doyle, A., *WWW Mapping Framework*, OGC Document #97-009, April 1997.

3. Elmasri R, Navathe S.B., (2011), *Database Systems: Models, Languages, Design and Application Programming*, Pearson, ISBN 10:0-13-214498-0

4. *ESRI Shapefile Technical Description*, An ESRI White Paper-July 1998

5. Fielding R. T. (2000), *Architectural Styles and the Design of Network-based Software Architectures*, http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf (visited 2013-11-20)

6. Gardels K., "*A Web Mapping Scenario*", OGC Document #98, January 1998

7. Hazzard E, (2011), *OpenLayers 2.10*: Beginner's Guide, Create, optimize, and deploy stunning cross-browser web maps with the OpenLayers JavaScript web-mapping library, Packt Publishing Ltd, Birmingham, UK, ISBN 978-1-849514-12-5

8. Howard M. Williams and Omar Dreza, *Combining Heterogeneous Spatial Data From Distributed Sources*, Shool of Math & Comp.Sc., Heriot-Watt Univ., Riccarton, Edinburgh, EH144ASUK: http://download.springer.com/static/pdf/410/chp%253A10.1007%252F3-540-26772-7_5.pdf (visited 2013-11-22)

9. Iacovella S., Youngblood B., (2013), *GeoServer Beginner's Guide: Share and edit geospatial data with this open source software server,* Packt Publishing Ltd, Birmingham, UK, ISBN 978-1-84951-668-6

10. Neteler M., Mitasova H., (2008), *Open Source GIS: A GRASS GIS Approach,* third edition, Springer science, United Kingdom, ISBN-13:978-0-387—35767-6

11. OpenGIS Consortium, *Request for Technology In Support of a Web Mapping Technology Testbed*, Oktober 1998

12. OpenGIS Consortium, *Request For Quotation and Call For Participation in the OGC Web Mapping Testbed Initial Operating Capability and Demonstration,* March 1999.

13. *OpenLayers.Layers.WMS*, http://dev.openlayers.org/docs/files/OpenLayers/Layer/WMS-js.html (visited 2013-12-12)

14. *PostGIS 2.1.2dev manual*, http://postgis.net/docs/manual-2.1/ (visited 2013-11-18)

15. *PostGIS: introduction*, http://workshops.boundlessgeo.com/postgis-intro/introduction.html (visited 2013-12-16)

16. *PostgreSQL*, http://www.postgresql.org/files/documentation/pdf/9.3/postgresql-9.3-A4.pdf (visited 2013-12-16)

17. *SharePoint products and technologies developer documentation,* http://msdn.microsoft.com/en-us/library/fp161348.aspx (visited 2013-12-11)

18. *SharePoint List*, http://office.microsoft.com/en-us/sharepoint-server-help/sharepoint-lists-i-an-introduction-RZ101820091.aspx (visited 2013-12-10)

19. *SharePoint List and Libraries*, http://msdn.microsoft.com/en-us/library/dd490727%28v=office.12%29.aspx (visited 2013-12-11)

20. *SharePoint SPList*, http://msdn.microsoft.com/en-us/library/microsoft.sharepoint.splist%28v=office.12%29.ASPX (visited 2013-11-25)

21. *ST_GeomFromText*, http://postgis.org/docs/ST_GeomFromText.html (visited 2013-10-5)

22. *SQL Server Express 2012*, http://technet.microsoft.com/en-us/library/ms130214.aspx (visited 2013-10-3)

23. *SQL-View*, http://docs.geoserver.org/stable/en/user/data/database/sqlview.html (visited 2013-11-20)

24. Walther S., (2004), *ASP.NET Unleashed,* second edition, Sams Publishing US, ISBN: 0-672-32542-X

25. *WMS Implementation Specification*, Open GIS Consortium Inc., 2012, http://www.opengeospatial.org/standards/wms (downloaded 2013-10-7)

DOWNLOADS

26. GeoServer, http://sourceforge.net/projects/geoserver/files/GeoServer/2.3.5/geoserver-2.3.5-war.zip/download (downloaded 2013-09-09)

27. Tomcat Apache, http://tomcat.apache.org (downloaded 2013-09-09)

8 Appendix

DBMSConnection

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data;
using Npgsql;
using GeoAPI.Geometries;
using GisSharpBlog.NetTopologySuite.Features;
using GisSharpBlog.NetTopologySuite.Geometries;
using GisSharpBlog.NetTopologySuite.IO;
using System.Collections;
using System.Data.SqlClient;


namespace DBMSConnection
{
    class ConnectToDatabases
    {
        private ArrayList allPaths;
        private ArrayList featureCollection;
        private List<ArrayList> allShapeFiles;
        private ShapefileDataReader shDataReader;

        //constructor
        public ConnectToDatabases()
        {
            allPaths = new ArrayList();
            allShapeFiles = new List<ArrayList>();
            featureCollection = new ArrayList();
            PrepareData();
            CreateCorrespondenceTableSQLExpress();
            ProcessData();
        }

        // this method fills the ArrayList allPaths with paths to *.shp
        public void PrepareData()
        {
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Points_100");
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Points_1k");
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Points_10k");
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Points_100k");
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Lines_100");
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Lines_1k");
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Lines_10k");
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Lines_100k");
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Polygons_100");
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Polygons_1k");
         allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Polygons_10k");
        allPaths.Add("C:\\Users\\SEMDAH\\Documents\\exjobb\\XJOBB\\Polygons_100k");
        }

        // this method goes through the ArrayList allPaths, reads each .shp file
        // with private method ReadShapeFile(String pPath):ArrayList
        // and adds this file into ArrayList allShapeFiles
        public void ProcessData()
        {
            ArrayList tmpArrayList = new ArrayList();
            allShapeFiles.Clear();
            foreach (String path in allPaths)
            {
```

```
            char[] delimiterChar = { ':', '\\' };
            string[] words = path.Split(delimiterChar);
            string tableName = words[words.Length - 1];
            tmpArrayList.Clear();
            CreateTablesSQLExpress(tableName);
            tmpArrayList = ReadShapeFile(path, tableName);
            allShapeFiles.Add(tmpArrayList);
        }
    }

    // the private method that reads the .shp
    private ArrayList ReadShapeFile(String pPath, String pTableName)
    {
        featureCollection.Clear();
        shDataReader = new ShapefileDataReader(pPath, new GeometryFactory());
        string typeOfGeometry = "";
        int length = 0;
        int counter = 0;
        Guid tmpGuid = Guid.NewGuid();
        String guidString = tmpGuid.ToString();
        ConnectToCorrespondenceTable(guidString, pTableName);

        while (shDataReader.Read())
        {
            GisSharpBlog.NetTopologySuite.Features.Feature feature =
                new GisSharpBlog.NetTopologySuite.Features.Feature();
            feature.Geometry = shDataReader.Geometry;
            typeOfGeometry = shDataReader.Geometry.GeometryType;
            GeoAPI.Geometries.IGeometry geometry = feature.Geometry;
            length = shDataReader.DbaseHeader.NumFields;
            string[] keys = new string[length];

            //goes into loop 2 times
            //1:st Guid
            //2:nd ORIG_FEED
            for (int i = 0; i < length; i++)
            {
                keys[i] = shDataReader.DbaseHeader.Fields[i].Name;
                feature.Attributes = new AttributesTable();
            }

            //goes into 2 times
            //1:st Guid value
            //2:nd ORIG_FEED value
            for (int j = 0; j < length; j++)
            {
                object val = new Object();
                val = shDataReader.GetValue(j);
                feature.Attributes.AddAttribute(keys[j], val);
            }
            featureCollection.Add(feature);
            string tmpGeometry = geometry.ToString();

            if (typeOfGeometry == "Point")
            {
                //tmpObject[1] returns ORIG_FEED for each specific point
                object[] tmpObject = feature.Attributes.GetValues();
                string origFeed = tmpObject[1].ToString();
                ConnectToPostgreSQL(tmpGuid, origFeed, tmpGeometry);
                ConnectToSQL(pTableName, tmpGuid, origFeed);
            }
            else if (typeOfGeometry == "MultiLineString" ||
                    typeOfGeometry == "Polygon")
            {
                string origFeed = "" + counter;
                ConnectToPostgreSQL(tmpGuid, origFeed, tmpGeometry);
                ConnectToSQL(pTableName, tmpGuid, origFeed);
            }
```

```
                counter++;
            }
            return featureCollection;
        }

        /*This method reads the information from the shapefile
            into PostgreSQL database*/
    public void ConnectToPostgreSQL(Guid pListGuid, String pItemGuid, String pGeom)
    {
        NpgsqlConnection connection = new NpgsqlConnection("Server=127.0.0.1;”
              +”Port=5432;User Id=postgres;Password=exjobbHT13;Database=postgres");
        string sqlGeom = "ST_GeomFromText('" + pGeom + "', 4326)";
        string tmpListGuid = "'" + pListGuid.ToString() + "'";
        string sql = string.Format("INSERT INTO spatial (ListGuid, ItemGuid, geom)"
                        +"VALUES({0},{1},{2})", tmpListGuid, pItemGuid, sqlGeom);
            NpgsqlCommand sqlCommand = new NpgsqlCommand(sql, connection);
            try
            {
                connection.Open();
                Int32 rowsaffected = sqlCommand.ExecuteNonQuery();
            }
            catch (Exception e)
            {
                System.Console.WriteLine("exception: {0}", e.ToString());
            }
            finally
            {
                connection.Close();
            }
     }

        /*This method creates an appropriate table for each shapefile*/
    public void CreateTablesSQLExpress(String pTableName)
    {
       System.Data.SqlClient.SqlConnection connection = null;
       string connectionString = @"Data Source=PCLUL10086\SQLEXPRESS;Initial
                                   Catalog=Meta;Integrated Security=True";
       string featureTableString = String.Format("CREATE TABLE {0} (ListGuid
                                   VARCHAR(50),ItemGuid VARCHAR(50), PRIMARY
                                   KEY(ListGuid, ItemGuid))", pTableName);
       using (connection = new
                   System.Data.SqlClient.SqlConnection(connectionString))
      {
        try
        {
          connection.Open();
          using (SqlCommand createFeatureTableCommand = new
                           SqlCommand(featureTableString, connection))
          {
              createFeatureTableCommand.ExecuteNonQuery();
          }
              connection.Close();
        }
        catch (Exception e)
        {
           System.Console.WriteLine(e.ToString());
        }
      }
    }


public void CreateCorrespondenceTableSQLExpress()
{
  System.Data.SqlClient.SqlConnection connection = null;
  string connectionString = @"Data Source=PCLUL10086\SQLEXPRESS;Initial
                              Catalog=Meta;Integrated Security=True";
  string tableCorrespondenceString = "CREATE TABLE TableCorrespondence(ListGuid
                              VARCHAR(50)PRIMARY KEY, ItemGuid VARCHAR(50))";
```

```
    using (connection = new System.Data.SqlClient.SqlConnection(connectionString))
    {
      try
      {
            connection.Open();
            using (SqlCommand createCorrespondenceTableCommand = new
                    SqlCommand(tableCorrespondenceString, connection))
            {
                createCorrespondenceTableCommand.ExecuteNonQuery();
            }
                connection.Close();
         }
         catch (Exception e)
         {
            System.Console.WriteLine(e.ToString());
         }
       }
     }

public void ConnectToCorrespondenceTable(String pGuidString, String pTableName)
{
  System.Data.SqlClient.SqlConnection connection = null;
  string connectionString = @"Data Source=PCLUL10086\SQLEXPRESS;Initial
                             Catalog=Meta;Integrated Security=True";

//opens a database connection with the property settings specified by the
//connectionString
using (connection = new System.Data.SqlClient.SqlConnection(connectionString))
{
  try
  {
    connection.Open();
    string insertString = "INSERT INTO TableCorrespondence VALUES(@ListGuid,
                           @TableName)";
    using (SqlCommand insertCommand = new SqlCommand(insertString, connection))
    {
      insertCommand.Parameters.Add(new SqlParameter("ListGuid", pGuidString));
      insertCommand.Parameters.Add(new SqlParameter("TableName", pTableName));
      Int32 rowsAffected = insertCommand.ExecuteNonQuery();
    }
     connection.Close();
  }
  catch (Exception e)
  {
     System.Console.WriteLine("exception: {0}", e.ToString());
  }
 }
}

/*This method write listGuid and itemGuid into a given table*/
public void ConnectToSQL(String pTableName, Guid pListGuid, String pItemGuid)
{
  System.Console.WriteLine("CONNECT TO SQLEXPRESS");
  string tmpListGuid = pListGuid.ToString();
  //Initializes a new instance of the SqlConnection class
  //when given a string that contains the connection string
  System.Data.SqlClient.SqlConnection connection = null;
  string connectionString = @"Data Source=PCLUL10086\SQLEXPRESS;Initial
                             Catalog=Meta;Integrated Security=True";

 //opens a database connection with the property settings
 //specified by the connectionString
 using (connection = new System.Data.SqlClient.SqlConnection(connectionString))
 {
   try
   {
     connection.Open()
     string tableName = String.Format("INSERT INTO {0}", pTableName);
```

```
            string insertString = tableName + " VALUES(@ListID, @ListItemID)";
            using (SqlCommand insertCommand = new SqlCommand(insertString, connection))
            {
              insertCommand.Parameters.Add(new SqlParameter("ListID", tmpListGuid));
               insertCommand.Parameters.Add(new SqlParameter("ListItemID", pItemGuid));
               Int32 rowsAffected = insertCommand.ExecuteNonQuery();
              }
              connection.Close();
             }
             catch (Exception e)
             {
                 System.Console.WriteLine("exception: {0}", e.ToString());
             }
            }
          }
        }

     public class Tester
     {
         static void Main(string[] args)
         {
             ConnectToDatabases connectionClass = new ConnectToDatabases();
         }
     }
}
```

Raster

Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="WebServices._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
     <script src="http://code.jquery.com/jquery-1.9.1.js"
             type = "text/javascript"></script>
     <script src="Scripts/jquery-1.4.1.js" type="text/javascript"></script>
     <script src="OpenLayers.js" type="text/javascript"></script>
     <script type="text/javascript">

       var map;
       var lat = 63.114155645298;
       var lon = 12.833434184876;
       var zoom = 5;
       var untiled;
       var format = "image/png";
       var counter = 0;
       var i = 0;
       var myCounter;
       function init() {

           var bounds = new OpenLayers.Bounds(
                12.6215390603863, 63.0654584041901,
                16.5880768069182, 63.1302372247887
           );

           var options = {
               maxExtent: bounds,
               maxResolution: 0.0154942880723902,
               projection: "EPSG:4326",
               units: 'm'
           };

           map = new OpenLayers.Map('map_element', options);
           untiled = new OpenLayers.Layer.WMS(
           "MD:points-Untiled","http://localhost:8080/geoserver/MD/wms",
            {
            LAYERS: 'MD:points',
            STYLES: '',
            format: format,
            CQL_FILTER: "lgstring='b3ef7444-09e2-451c-9bec-1e50f19a3592'"
            },
            {
                isBaseLayer: true,
                ratio: 1,
                opacity: 0.5,
               singleTile: true,
                yx: { 'EPSG:4326': true }
            }
            );

          map.addLayers([untiled]);
          map.addControl(new OpenLayers.Control.LayerSwitcher());
          if(!map.getCenter()){
          map.zoomToExtent(bounds);
           }
       }
```

```
//testOneLayer
function testOneLayer() {
    getLayer();
    counter = 0;
    myCounter = setInterval(function () { getLayer() }, 60000);
 }

function getLayer() {
   $("input[name=figure]").each(function (index) {
       if ($(this)[0].checked) {
           var quFeatures = $(this)[0].id;
           var quLayer = quFeatures.split("_");
           var image_layer = new OpenLayers.Layer.WMS(
               "MD:" + quFeatures, 'service.ashx?layers=MD:' + quLayer[0] +
                '&cql_filter=' + quFeatures,
              { transparent: true },
              {
                  isBaseLayer: false,
                  ratio: 1,
                  singleTile: true,
                  yx: { 'EPSG:4326': true }
              }
            );
                 map.addLayer(image_layer);
           }
         });
      counter++;
    if (counter == 10) { clearInterval(myCounter); }
  }

  //testOneLayer
  function testWithLimit(){
      getLayerWithLimit();
      counter = 0;
      myCounter = setInterval(function () { getLayerWithLimit() }, 5000);
  }

  function getLayerWithLimit() {
      $("input[name=figure]").each(function (index) {
          if ($(this)[0].checked) {
              var quFeatures = $(this)[0].id;
              var quLayer = quFeatures.split("_");
              var image_layer = new OpenLayers.Layer.WMS(
              "MD:" + quFeatures, 'service.ashx?layers=MD:' + quLayer[0]
                    +'_limit'+ '&cql_filter=' + quFeatures,
              { transparent: true },
              {
                  isBaseLayer: false,
                  ratio: 1,
                  singleTile: true,
                  yx: { 'EPSG:4326': true }
              }
            );
              map.addLayer(image_layer);
          }
      });
     counter++;
     if (counter == 10) { clearInterval(myCounter); }
  }


  //testOneZoome
  function testOneZoom(value) {
      getOneZoom(value);
      counter = 0;
      myCounter = setInterval(function () { getOneZoom(value) }, 90000);
```

```
        }

        function getOneZoom(value) {
            $("#feature" + value).each(function (index) {
                $((this).children).each(function (index) {
                    var quFeatures = $(this).text();
                    var quLayer = $(this).text().split("_");
                    var image_layer = new OpenLayers.Layer.WMS(
                         "MD:" + quFeatures, 'service.ashx?layers=MD:' + quLayer[0]
                               + '&cql_filter=' + quFeatures,
                         { transparent: true },
                         {
                             isBaseLayer: false,
                             ratio: 1,
                             singleTile: true,
                             yx: { 'EPSG:4326': true }
                         }
                     );
                    map.addLayer(image_layer);
                });
            });
           counter++;
           if (counter == 10) { clearInterval(myCounter); }
         }


        //testAllLayers
        function testAllLayers() {
            $("td").each(function (index) {
                var quFeatures = $(this).text();
                var quLayer = $(this).text().split("_");
                var image_layer = new OpenLayers.Layer.WMS(
                     "MD:" + quFeatures, 'service.ashx?layers=MD:' + quLayer[0] +
                           '&cql_filter=' + quFeatures,
                     {transparent: true },
                     {
                         isBaseLayer: false,
                         ratio: 1,
                         singleTile: true,
                         yx: { 'EPSG:4326': true }
                     }
                 );
                map.addLayer(image_layer);
            });
        }
    </script>
</head>
<body  onload='init();'>
<form id="form1" runat="server">
<div id = "testContainer" style='width:1340px; height: 220px; border-style:solid;
     border-width: 1px; padding: 5px'>
    <div id='radio_button_group' style='width:350px; height: 210px; border-
         style:solid; border-width: 1px; padding: 5px; float:left'>
       <h2>Test one layer</h2>
        <table>
           <tr id = "feature100">
             <td><input type='radio' name='figure' id='points_100' value =
                   'points_100' checked='checked'autocomplete="off"/>points_100</td>
             <td><input type="radio" name="figure" id="lines_100" value =
                    "lines_100" autocomplete="off"/>lines_100 </td>
             <td><input type="radio" name="figure" id="polygons_100" value =
                    "polygons_100" autocomplete="off"/>polygons_100</td>
           </tr>
           <tr id="feature1000">
             <td><input type="radio" name="figure" id="points_1k" value =
                    "points_1k" autocomplete="off" />points_1k</td>
             <td><input type="radio" name="figure" id="lines_1k" value = "lines_1k"
                     autocomplete="off" />lines_1k</td>
```

```
            <td><input type="radio" name="figure" id="polygons_1k" value =
                  "polygons_1k"autocomplete="off"/>polygons_1k</td>
          </tr>
          <tr id="feature10000">
            <td><input type="radio" name="figure" id="points_10k" value =
                 "points_10k" autocomplete="off" />points_10k</td>
            <td><input type="radio" name="figure" id="lines_10k" value =
                 "lines_10k" autocomplete="off"/>lines_10k</td>
            <td><input type="radio" name="figure" id="polygons_10k" value =
                  "polygons_10k" autocomplete="off"/>polygons_10k</td>
          </tr>
          <tr id="feature100000">
            <td><input type="radio" name="figure" id="points_100k" value =
                "points_100k"  autocomplete="off"/>points_100k </td>
            <td><input type="radio" name="figure" id="lines_100k" value =
                "lines_100k" autocomplete="off" />lines_100k</td>
            <td><input type="radio" name="figure" id="polygons_100k" value =
                 "polygons_100k" autocomplete="off"/>polygons_100k</td>
          </tr>
      </table>
      <p><input type="button" onclick="testOneLayer();" value="Test One Layer" />
      <input type="button" onclick="testWithLimit();" value="Manual test" /></p>
  </div>
  <div id = "testEachZoom" style='width:600px; height: 210px; border-style:solid;
        border-width: 1px; padding: 5px; float:left'>
     <h2>Test one zoom</h2>
     <p>This part of the test sends 3 requests to the server (points, lines,
        polygons) <br /> and puts each responce layer on the map as soon it is
        received.<br /><br /><br /><br /></p>
     <p>
     <input type="button" onclick="testOneZoom(this.name);" name = "100"
            value="Test 100 Features" />
     <input type="button" onclick="testOneZoom(this.name);" name = "1000"
            value="Test 1000 Features" />
     <input type="button" onclick="testOneZoom(this.name);" name = "10000"
            value="Test 10000 Features" />
     <input type="button" onclick="testOneZoom(this.name);" name = "100000"
            value="Test 100000 Features" />
     </p>
 </div>
 <div id = "testAllLayers" style='width:350px; height: 210px; border-style:solid;
      border-width: 1px; padding: 5px; float:left';>
        <h2>Test all layers</h2>
        <p>This part of the test sends 12 requests to the server (one request for
           each layer) and puts each responce layer on the map as soon it is
           received.<br /> This test can take some time. <br /><br /></p>
        <p><input type="button" onclick="testAllLayers();" value="Test All Layers"
           /> </p>
  </div>
</div>
<br />
<br />
  <div id='map_element' style='width:1350px; height:350px; border: 1px solid black;
clear: both; position: relative; float:left'></div>
</form>
</body>
</html>
```

Service.ashx

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Data;
using Npgsql;
using GeoAPI.Geometries;
using GisSharpBlog.NetTopologySuite.Features;
using GisSharpBlog.NetTopologySuite.Geometries;
using GisSharpBlog.NetTopologySuite.IO;
using System.Collections;
using System.Data.SqlClient;
using System.Net;
using System.IO;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Drawing;
using System.Diagnostics;
namespace WebServices
{

    /* This class simulates the server that generates response
     * to the client
     */
    public class Service : IHttpHandler
    {

        HttpContext ctx;
        String requestedFeature;
        String featureGuid;
        System.Drawing.Image responsePNG;
        byte[] rspPNG;
        List<List<string>> allLayersTimeLog = new List<List<string>>();
        List<string> layerTimeLog = new List<string>();

        // method is inherited from IHttpHandler
        public void ProcessRequest(HttpContext context)
        {
            Stopwatch stopWatch = new Stopwatch();
            stopWatch.Start();
            ctx = context;
            requestedFeature = ctx.Request["cql_filter"].Trim();
            layerTimeLog.Add(requestedFeature);
            layerTimeLog.Add(stopWatch.ElapsedMilliseconds.ToString());
            GetImage();
            context.Response.ContentType = "image/png";
            context.Response.BinaryWrite(rspPNG);
            stopWatch.Stop();
            layerTimeLog.Add(stopWatch.ElapsedMilliseconds.ToString());
            TimeSpan timeSpan = stopWatch.Elapsed;
            allLayersTimeLog.Add(layerTimeLog);
            writeToFile(layerTimeLog, timeSpan);

        }

        /* This method asks the SQL EXpress about guid
         * and asks the Geoserver to generate the layer that
         * is associated with the specified guid for
         * the requested number of features*/
        void GetImage()
        {
            featureGuid = GetTableGuid();
            WMSParams wmsParams = CreateWMSObj();
            GetImageFromGeoServer();
        }

        /*This method builds the WMS request*/
```

```
private WMSParams CreateWMSObj()
{
    WMSParams lWmsParams = new
                WMSParams(ctx.Request["layers"].Trim(),featureGuid);
    wmsParams = lWmsParams.UrlString;
    return lWmsParams;
}

/* This method connects the Service.ashx to the SQL Express
 * and asks the SQL Expresss about guid of the table with
 * the requested numbers of features*/
private string GetTableGuid()
{
    /* ConfigurationManager provides access to configuration files for the
     * client applications. To read a section from a configuration file use
     * an appropriate class (ConnectionsStrings)This class performs read-
     * only operations, use a single cached instance of the configuration,
     * and are multithread aware.
     * ConnectionStrings property gets the ConnectionStringSection data
     * for the current application's default configuration
     * */

    Stopwatch sqlExpressWatch = new Stopwatch();
    sqlExpressWatch.Start();
    layerTimeLog.Add(sqlExpressWatch.ElapsedMilliseconds.ToString());
    var connectionString =
    System.Configuration.ConfigurationManager.ConnectionStrings["MetaConnec
    tionString"];

    //Initializes a new instance of the SqlConnection class when given a
    //string that contains the connection string
    System.Data.SqlClient.SqlConnection connection = null;
    String sqlString = String.Format("SELECT TableGuid FROM
           TableCorrespondence WHERE TableName='{0}'", requestedFeature);
    SqlCommand command= null;
    SqlDataReader reader = null;
    String queriedGuid = "";
    try
    {
     connection = new
    System.Data.SqlClient.SqlConnection(connectionString.ConnectionString);
        //opens a database connection with the property settings specified
        //by the connectionString
        connection.Open();
        command = new SqlCommand(sqlString,connection);
        reader = command.ExecuteReader();

        while (reader.Read())
        {
            for (int i = 0; i < reader.FieldCount; i++ )
            {
                queriedGuid = reader[i].ToString();
            }
        }
        connection.Close();
    }
    catch(Exception e)
    {
        Console.WriteLine(e.Message);
    }

    sqlExpressWatch.Stop();
    layerTimeLog.Add(sqlExpressWatch.ElapsedMilliseconds.ToString());
    return queriedGuid;
}

/* This method sends the request to GeoServer and generates the responce*/
private void GetImageFromGeoServer()
```

```
        {
            Stopwatch geoserverWatch = new Stopwatch();
            geoserverWatch.Start();
            layerTimeLog.Add(geoserverWatch.ElapsedMilliseconds.ToString());
        try
            {
                WebRequest request = WebRequest.Create(wmsParams);
                HttpWebResponse response = (HttpWebResponse)request.GetResponse();
                Stream dataStream = response.GetResponseStream();
                responsePNG = System.Drawing.Image.FromStream(dataStream);
                using (MemoryStream memoryStream = new MemoryStream()) {
                        responsePNG.Save(memoryStream,
                        System.Drawing.Imaging.ImageFormat.Png);
                    rspPNG = memoryStream.ToArray();

                }
                dataStream.Close();
                response.Close();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }

        geoserverWatch.Stop();
        layerTimeLog.Add(geoserverWatch.ElapsedMilliseconds.ToString());
        }

        // Inherited proerty from the IHttpHandler interface.
        // This property gets the value indicating wheter another request can
        // use the IHttpHandler instance.
        // Because this is a syncronous handler, return False for
        // the isReusable property so that the handler is not pooled, the
        //IHttpHandlerFactory object can not put the handler in the pool and reuse
        //it to increase performance. If the handler can not be pooled the, the
        //factory must create a new instance of the handler every time that
        //the handler is needed.
        public bool IsReusable
        {
            get{ return false;}
        }

        public string wmsParams { get; set; }

        /* This method writes information into the TimeLog*/
        private void writeToFile(List<string> pLayerTimeLog, TimeSpan pTimeSpan)
        {
            using (System.IO.StreamWriter file =
            File.AppendText(@"C:\Users\SEMDAH\Documents\exjobb\Raster\Raster\WebSer
            vices\timeLog.txt"))
            {
                foreach(string tmp in pLayerTimeLog)
                {
                    file.Write(tmp +"           ");
                }
                     string elapsedTime =
                        String.Format("{0:00}:{1:00}:{2:00}.{3:00}",pTimeSpan.Hours
                        ,pTimeSpan.Minutes,pTimeSpan.Seconds,pTimeSpan.Milliseconds
                        /10);
                        file.WriteLine(elapsedTime);
                        file.WriteLine();
            }
        }
    }
}
```

WMSParams.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace WebServices
{
    public class WMSParams
    {
        string urlRequest;
        List<string> parameters = new List<string>();
        public List<string> Params { get { return parameters; } }

        private string urlString =
        "http://localhost:8080/geoserver/MD/wms?service=WMS&version=1.1.0&request=G
        etMap&styles=&bbox=12.6215390603863,63.0654584041901,16.5880768069182,63.13
        02372247887&width=20206&height=330&srs=EPSG:4326&format=image%2Fpng";

        //Property
        public  string UrlString {
            get{ return urlString; }
        }

        //Constructor
        public WMSParams(String layers, String cql_filter)
        {
             urlString += "&layers=" + layers +
            "&cql_filter=lgstring%3D%27"+cql_filter +"%27";
        }
    }
}
```

Web.config

```
<?xml version="1.0"?>
<configuration>
  <appSettings/>
  <system.web>
    <compilation debug="true" targetFramework="4.0"/>
    <!--
            The <authentication> section enables configuration
            of the security authentication mode used by
            ASP.NET to identify an incoming user.
        -->
    <authentication mode="Windows"/>
    <!--
            The <customErrors> section enables configuration
            of what to do if/when an unhandled error occurs
            during the execution of a request. Specifically,
            it enables developers to configure html error pages
            to be displayed in place of a error stack trace.

        <customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
            <error statusCode="403" redirect="NoAccess.htm" />
            <error statusCode="404" redirect="FileNotFound.htm" />
        </customErrors>
        -->
    <pages controlRenderingCompatibilityVersion="3.5" clientIDMode="AutoID"/>
  </system.web>
  <!--
        The system.webServer section is required for running ASP.NET AJAX under
Internet
        Information Services 7.0.  It is not necessary for previous version of IIS.
    -->
  <connectionStrings>
    <add name="MetaConnectionString" connectionString="Data
Source=PCLUL10086\SQLEXPRESS;Initial Catalog=Meta;Integrated Security=True"
```

```
      providerName="System.Data.SqlClient" />
    <add name="SpatialConnectionString" connectionString="User
Id=postgres;Password=exjobbHT13;Host=localhost;Database=Spatial;Persist Security
Info=True;Initial Schema=public"
      providerName="Devart.Data.PostgreSql" />
  </connectionStrings>
</configuration>
```

TimeLog

LAYER	START	FINISCH	TIME TO RENDER
			TEST 1
points_100	0	1046	00:00:01.04
points_100	0	1213	00:00:01.21
points_100	0	1051	00:00:01.05
points_100	0	1047	00:00:01.04
points_100	0	1051	00:00:01.05
points_100	0	1218	00:00:01.21
points_100	0	1048	00:00:01.04
points_100	0	1046	00:00:01.04
points_100	0	1051	00:00:01.05
points_100	0	1236	00:00:01.23
points_100	0	1040	00:00:01.04
points_100	0	1045	00:00:01.04
points_100	0	1039	00:00:01.03
points_100	0	1211	00:00:01.21
points_100	0	1044	00:00:01.04
points_100	0	1054	00:00:01.05
points_100	0	1043	00:00:01.04
points_100	0	1204	00:00:01.20
points_100	0	1053	00:00:01.05
points_100	0	1038	00:00:01.03
			TEST 1
points_1k	0	1144	00:00:01.14
points_1k	0	1132	00:00:01.13
points_1k	0	1143	00:00:01.14
points_1k	0	1310	00:00:01.31
points_1k	0	1127	00:00:01.12
points_1k	0	1133	00:00:01.13
points_1k	0	1136	00:00:01.13
points_1k	0	1295	00:00:01.29
points_1k	0	1132	00:00:01.13
points_1k	0	1134	00:00:01.13
points_1k	0	1137	00:00:01.13
points_1k	0	1297	00:00:01.29
points_1k	0	1132	00:00:01.13
points_1k	0	1132	00:00:01.13
points_1k	0	1128	00:00:01.12
points_1k	0	1297	00:00:01.29
points_1k	0	1128	00:00:01.12
points_1k	0	1129	00:00:01.12
points_1k	0	1126	00:00:01.12
points_1k	0	1305	00:00:01.30
			TEST 1
points_10k	0	1639	00:00:01.63
points_10k	0	1638	00:00:01.63
points_10k	0	1627	00:00:01.62
points_10k	0	1797	00:00:01.79
points_10k	0	1625	00:00:01.62
points_10k	0	1615	00:00:01.61
points_10k	0	1625	00:00:01.62
points_10k	0	1791	00:00:01.79
points_10k	0	1625	00:00:01.62
points_10k	0	1616	00:00:01.61
points_10k	0	1623	00:00:01.62
points_10k	0	1808	00:00:01.80
points_10k	0	1620	00:00:01.62
points_10k	0	1619	00:00:01.61
points_10k	0	1626	00:00:01.62
points_10k	0	1796	00:00:01.79
points_10k	0	1623	00:00:01.62
points_10k	0	1617	00:00:01.61
points_10k	0	1624	00:00:01.62
points_10k	0	1785	00:00:01.78
			TEST 1

LAYER	START	FINISCH	TIME TO RENDER
points_100k	0	5532	00:00:05.53
points_100k	0	5503	00:00:05.50
points_100k	0	5727	00:00:05.72
points_100k	0	5442	00:00:05.44
points_100k	0	5663	00:00:05.66
points_100k	0	5437	00:00:05.43
points_100k	0	5791	00:00:05.79
points_100k	0	5449	00:00:05.44
points_100k	0	5513	00:00:05.51
points_100k	0	5617	00:00:05.61
points_100k	0	5429	00:00:05.42
points_100k	0	5691	00:00:05.69
points_100k	0	5514	00:00:05.51
points_100k	0	5706	00:00:05.70
points_100k	0	5495	00:00:05.49
points_100k	0	5683	00:00:05.68
points_100k	0	5442	00:00:05.44
points_100k	0	5654	00:00:05.65
points_100k	0	5446	00:00:05.44
points_100k	0	5681	00:00:05.68
			TEST 1
lines_100	0	1034	00:00:01.03
lines_100	0	1022	00:00:01.02
lines_100	0	1033	00:00:01.03
lines_100	0	1194	00:00:01.19
lines_100	0	1022	00:00:01.02
lines_100	0	1026	00:00:01.02
lines_100	0	1024	00:00:01.02
lines_100	0	1197	00:00:01.19
lines_100	0	1021	00:00:01.02
lines_100	0	1033	00:00:01.03
lines_100	0	1023	00:00:01.02
lines_100	0	1190	00:00:01.19
lines_100	0	1028	00:00:01.02
lines_100	0	1034	00:00:01.03
lines_100	0	1030	00:00:01.03
lines_100	0	1186	00:00:01.18
lines_100	0	1018	00:00:01.01
lines_100	0	1028	00:00:01.02
lines_100	0	1023	00:00:01.02
lines_100	0	1188	00:00:01.18
			TEST 1
lines_1k	0	1131	00:00:01.13
lines_1k	0	1125	00:00:01.12
lines_1k	0	1125	00:00:01.12
lines_1k	0	1304	00:00:01.30
lines_1k	0	1127	00:00:01.12
lines_1k	0	1123	00:00:01.12
lines_1k	0	1127	00:00:01.12
lines_1k	0	1295	00:00:01.29
lines_1k	0	1123	00:00:01.12
lines_1k	0	1132	00:00:01.13
lines_1k	0	1122	00:00:01.12
lines_1k	0	1317	00:00:01.31
lines_1k	0	1128	00:00:01.12
lines_1k	0	1126	00:00:01.12
lines_1k	0	1132	00:00:01.13
lines_1k	0	1297	00:00:01.29
lines_1k	0	1132	00:00:01.13
lines_1k	0	1128	00:00:01.12
lines_1k	0	1121	00:00:01.12
			TEST 1
lines_10k	0	1731	00:00:01.73
lines_10k	0	1560	00:00:01.56
lines_10k	0	1543	00:00:01.54

lines_10k 0 1552 00:00:01.55
lines_10k 0 1719 00:00:01.71
lines_10k 0 1548 00:00:01.54
lines_10k 0 1541 00:00:01.54
lines_10k 0 1546 00:00:01.54
lines_10k 0 1712 00:00:01.71
lines_10k 0 1546 00:00:01.54
lines_10k 0 1545 00:00:01.54
lines_10k 0 1545 00:00:01.54
lines_10k 0 1710 00:00:01.71
lines_10k 0 1547 00:00:01.54
lines_10k 0 1540 00:00:01.54
lines_10k 0 1545 00:00:01.54
lines_10k 0 1715 00:00:01.71
lines_10k 0 1546 00:00:01.54
lines_10k 0 1542 00:00:01.54
lines_10k 0 1542 00:00:01.54
____TEST 1
lines_100k 0 5515 00:00:05.51
lines_100k 0 5497 00:00:05.49
lines_100k 0 5526 00:00:05.52
lines_100k 0 5577 00:00:05.57
lines_100k 0 5536 00:00:05.53
lines_100k 0 5550 00:00:05.55
lines_100k 0 5547 00:00:05.54
lines_100k 0 5540 00:00:05.54
lines_100k 0 5583 00:00:05.58
lines_100k 0 5537 00:00:05.53
lines_100k 0 5545 00:00:05.54
lines_100k 0 5501 00:00:05.50
lines_100k 0 5501 00:00:05.50
lines_100k 0 5506 00:00:05.50
lines_100k 0 5500 00:00:05.50
lines_100k 0 5511 00:00:05.51
lines_100k 0 5499 00:00:05.49
lines_100k 0 5585 00:00:05.58
lines_100k 0 5525 00:00:05.52
lines_100k 0 5537 00:00:05.53
____TEST 1
polygons_100 0 1014 00:00:01.01
polygons_100 0 1010 00:00:01.01
polygons_100 0 1013 00:00:01.01
polygons_100 0 1018 00:00:01.01
polygons_100 0 1018 00:00:01.01
polygons_100 0 1012 00:00:01.01
polygons_100 0 1014 00:00:01.01
polygons_100 0 1018 00:00:01.01
polygons_100 0 1009 00:00:01.00
polygons_100 0 1013 00:00:01.01
polygons_100 0 1011 00:00:01.01
polygons_100 0 1013 00:00:01.01
polygons_100 0 1009 00:00:01.00
polygons_100 0 1022 00:00:01.02
polygons_100 0 1017 00:00:01.01
polygons_100 0 1028 00:00:01.02
polygons_100 0 1018 00:00:01.01
polygons_100 0 1013 00:00:01.01
polygons_100 0 1009 00:00:01.00
polygons_100 0 1022 00:00:01.02
____TEST 1
polygons_1k 0 1079 00:00:01.07
polygons_1k 0 1064 00:00:01.06
polygons_1k 0 1072 00:00:01.07
polygons_1k 0 1063 00:00:01.06
polygons_1k 0 1063 00:00:01.06
polygons_1k 0 1067 00:00:01.06
polygons_1k 0 1079 00:00:01.07
polygons_1k 0 1083 00:00:01.08
polygons_1k 0 1068 00:00:01.06
polygons_1k 0 1069 00:00:01.06
polygons_1k 0 1077 00:00:01.07
polygons_1k 0 1075 00:00:01.07
polygons_1k 0 1066 00:00:01.06
polygons_1k 0 1068 00:00:01.06
polygons_1k 0 1084 00:00:01.08
polygons_1k 0 1075 00:00:01.07
polygons_1k 0 1060 00:00:01.06
polygons_1k 0 1067 00:00:01.06
polygons_1k 0 1079 00:00:01.07
polygons_1k 0 1066 00:00:01.06
____TEST 1
polygons_10k 0 1554 00:00:01.55
polygons_10k 0 1555 00:00:01.55
polygons_10k 0 1721 00:00:01.72
polygons_10k 0 1551 00:00:01.55
polygons_10k 0 1552 00:00:01.55
polygons_10k 0 1561 00:00:01.56
polygons_10k 0 1722 00:00:01.72
polygons_10k 0 1554 00:00:01.55
polygons_10k 0 1556 00:00:01.55
polygons_10k 0 1550 00:00:01.55
polygons_10k 0 1709 00:00:01.70
polygons_10k 0 1553 00:00:01.55
polygons_10k 0 1550 00:00:01.55
polygons_10k 0 1546 00:00:01.54
polygons_10k 0 1725 00:00:01.72
polygons_10k 0 1553 00:00:01.55
polygons_10k 0 1560 00:00:01.56
polygons_10k 0 1554 00:00:01.55
polygons_10k 0 1716 00:00:01.71
polygons_10k 0 1548 00:00:01.54
____TEST 1
polygons_100k 0 6559 00:00:06.55
polygons_100k 0 6749 00:00:06.74
polygons_100k 0 6772 00:00:06.77
polygons_100k 0 6778 00:00:06.77
polygons_100k 0 6736 00:00:06.73
polygons_100k 0 6745 00:00:06.74
polygons_100k 0 6743 00:00:06.74
polygons_100k 0 6787 00:00:06.78
polygons_100k 0 6743 00:00:06.74
polygons_100k 0 6754 00:00:06.75
polygons_100k 0 6753 00:00:06.75
polygons_100k 0 6746 00:00:06.74
polygons_100k 0 6743 00:00:06.74
polygons_100k 0 6773 00:00:06.77
polygons_100k 0 6755 00:00:06.75
polygons_100k 0 6763 00:00:06.76
polygons_100k 0 6741 00:00:06.74
polygons_100k 0 6756 00:00:06.75
polygons_100k 0 6734 00:00:06.73
polygons_100k 0 6825 00:00:06.82

____TEST 2 ZOOM____

________________ZOOM 100
____1
points_100 0 1205 00:00:01.20
polygons_100 0 1223 00:00:01.22
lines_100 0 1263 00:00:01.26
points_100 0 1188 00:00:01.18
polygons_100 0 1362 00:00:01.36
lines_100 0 1393 00:00:01.39
____2
points_100 0 1350 00:00:01.35
polygons_100 0 1361 00:00:01.36
lines_100 0 1366 00:00:01.36
points_100 0 1381 00:00:01.38
lines_100 0 1332 00:00:01.33

polygons_100	0	1381	00:00:01.38

____3

lines_100	0	1326	00:00:01.32
points_100	0	1339	00:00:01.33
polygons_100	0	1335	00:00:01.33
polygons_100	0	1359	00:00:01.35
points_100	0	1325	00:00:01.32
lines_100	0	1337	00:00:01.33

____4

points_100	0	1319	00:00:01.31
polygons_100	0	1342	00:00:01.34
lines_100	0	1350	00:00:01.35
points_100	0	1313	00:00:01.31
lines_100	0	1183	00:00:01.18
polygons_100	0	1229	00:00:01.22

____5

points_100	0	1131	00:00:01.13
lines_100	0	1374	00:00:01.37
polygons_100	0	1397	00:00:01.39
points_100	0	1347	00:00:01.34
lines_100	0	1339	00:00:01.33
polygons_100	0	1200	00:00:01.20

____6

points_100	0	1332	00:00:01.33
polygons_100	0	1333	00:00:01.33
lines_100	0	1342	00:00:01.34
points_100	0	1319	00:00:01.31
polygons_100	0	1335	00:00:01.33
lines_100	0	1344	00:00:01.34

____7

points_100	0	1311	00:00:01.31
polygons_100	0	1319	00:00:01.31
lines_100	0	1330	00:00:01.33
points_100	0	1325	00:00:01.32
polygons_100	0	1365	00:00:01.36
lines_100	0	1339	00:00:01.33

____8

polygons_100	0	1323	00:00:01.32
points_100	0	1337	00:00:01.33
lines_100	0	1344	00:00:01.34
points_100	0	1355	00:00:01.35
polygons_100	0	1306	00:00:01.30
lines_100	0	1393	00:00:01.39

____9

lines_100	0	1299	00:00:01.29
polygons_100	0	1292	00:00:01.29
points_100	0	1324	00:00:01.32
points_100	0	1328	00:00:01.32
lines_100	0	1316	00:00:01.31
polygons_100	0	1319	00:00:01.31

____10

points_100	0	1313	00:00:01.31
lines_100	0	1323	00:00:01.32
polygons_100	0	1301	00:00:01.30
polygons_100	0	1166	00:00:01.16
points_100	0	1351	00:00:01.35
lines_100	0	1376	00:00:01.37

__________________zoom 1000_____________________________________ZOOM 1000______________________

____1

polygons_1k	0	1378	00:00:01.37
lines_1k	0	1439	00:00:01.43
points_1k	0	1459	00:00:01.45
points_1k	0	1494	00:00:01.49
polygons_1k	0	1453	00:00:01.45
lines_1k	0	1489	00:00:01.48

____2

polygons_1k	0	1445	00:00:01.44
lines_1k	0	1515	00:00:01.51
points_1k	0	1535	00:00:01.53
points_1k	0	1649	00:00:01.64
polygons_1k	0	1502	00:00:01.50
lines_1k	0	1656	00:00:01.65

____3

polygons_1k	0	1396	00:00:01.39
points_1k	0	1434	00:00:01.43
lines_1k	0	1452	00:00:01.45
polygons_1k	0	1536	00:00:01.53
lines_1k	0	1588	00:00:01.58
points_1k	0	1353	00:00:01.35

____4

polygons_1k	0	1385	00:00:01.38
points_1k	0	1479	00:00:01.47
lines_1k	0	1500	00:00:01.50
points_1k	0	1647	00:00:01.64
lines_1k	0	1468	00:00:01.46
polygons_1k	0	1385	00:00:01.38

____5

polygons_1k	0	1633	00:00:01.63
lines_1k	0	1712	00:00:01.71
points_1k	0	1723	00:00:01.72
lines_1k	0	1332	00:00:01.33
points_1k	0	1627	00:00:01.62
polygons_1k	0	1644	00:00:01.64

____6

polygons_1k	0	1385	00:00:01.38
points_1k	0	1412	00:00:01.41
lines_1k	0	1417	00:00:01.41
polygons_1k	0	1345	00:00:01.34
points_1k	0	1591	00:00:01.59
lines_1k	0	1362	00:00:01.36

____7

polygons_1k	0	1376	00:00:01.37
points_1k	0	1454	00:00:01.45
lines_1k	0	1429	00:00:01.42
points_1k	0	1340	00:00:01.34
lines_1k	0	1174	00:00:01.17
polygons_1k	0	1116	00:00:01.11

____8

polygons_1k	0	1375	00:00:01.37
points_1k	0	1397	00:00:01.39
lines_1k	0	1429	00:00:01.42
points_1k	0	1595	00:00:01.59
lines_1k	0	1468	00:00:01.46
polygons_1k	0	1351	00:00:01.35

____9

polygons_1k	0	1543	00:00:01.54
lines_1k	0	1594	00:00:01.59
points_1k	0	1614	00:00:01.61
points_1k	0	1570	00:00:01.57
polygons_1k	0	1472	00:00:01.47
lines_1k	0	1561	00:00:01.56

____10

polygons_1k	0	1546	00:00:01.54
lines_1k	0	1566	00:00:01.56
points_1k	0	1592	00:00:01.59
points_1k	0	1585	00:00:01.58
lines_1k	0	1641	00:00:01.64
polygons_1k	0	1330	00:00:01.33

____________________________zoom 10000________________________ZOOM 1000________________________________

_______________________________________1

lines_10k 0 2401 00:00:02.40
polygons_10k 0 2488 00:00:02.48
points_10k 0 2802 00:00:02.80
lines_10k 0 1668 00:00:01.66

polygons_10k 0 1808 00:00:01.80
points_10k 0 1655 00:00:01.65

_______________________________________2

lines_10k 0 2402 00:00:02.40
polygons_10k 0 2432 00:00:02.43
points_10k 0 2782 00:00:02.78
lines_10k 0 2285 00:00:02.28
polygons_10k 0 2374 00:00:02.37
points_10k 0 2678 00:00:02.67

_______________________________________3

lines_10k 0 2362 00:00:02.36
polygons_10k 0 2389 00:00:02.38
points_10k 0 2647 00:00:02.64
lines_10k 0 2424 00:00:02.42
polygons_10k 0 2482 00:00:02.48
points_10k 0 2815 00:00:02.81

_______________________________________4

lines_10k 0 2375 00:00:02.37
polygons_10k 0 2419 00:00:02.41
points_10k 0 2766 00:00:02.76
points_10k 0 1892 00:00:01.89
lines_10k 0 1890 00:00:01.89
polygons_10k 0 1866 00:00:01.86

_______________________________________5

lines_10k 0 2589 00:00:02.58
polygons_10k 0 2628 00:00:02.62
points_10k 0 2926 00:00:02.92
polygons_10k 0 2382 00:00:02.38
lines_10k 0 2319 00:00:02.31
points_10k 0 2618 00:00:02.61

_______________________________________6

lines_10k 0 2444 00:00:02.44
polygons_10k 0 2483 00:00:02.48
points_10k 0 2846 00:00:02.84
lines_10k 0 1925 00:00:01.92
points_10k 0 2087 00:00:02.08
polygons_10k 0 1599 00:00:01.59

_______________________________________7

lines_10k 0 2202 00:00:02.20
polygons_10k 0 2430 00:00:02.43
points_10k 0 2755 00:00:02.75
points_10k 0 2141 00:00:02.14
lines_10k 0 2319 00:00:02.31
polygons_10k 0 2339 00:00:02.33

_______________________________________8

points_10k 0 2130 00:00:02.13
lines_10k 0 2269 00:00:02.26
polygons_10k 0 2366 00:00:02.36
lines_10k 0 2331 00:00:02.33
polygons_10k 0 2526 00:00:02.52
points_10k 0 2297 00:00:02.29

_______________________________________9

lines_10k 0 2514 00:00:02.51
polygons_10k 0 2566 00:00:02.56
points_10k 0 2894 00:00:02.89
lines_10k 0 2319 00:00:02.31
polygons_10k 0 2349 00:00:02.34
points_10k 0 2618 00:00:02.61

_______________________________________10

lines_10k 0 2284 00:00:02.28
polygons_10k 0 2384 00:00:02.38
points_10k 0 2699 00:00:02.69
lines_10k 0 2166 00:00:02.16
points_10k 0 2370 00:00:02.37
polygons_10k 0 2168 00:00:02.16

______________________zoom 100000________________________________ZOOM 100000________________________

_______________________________________1

lines_100k 0 29315 00:00:29.31
points_100k 0 31216 00:00:31.21
polygons_100k 0 31709 00:00:31.70
lines_100k 0 7601 00:00:07.60
polygons_100k 0 9424 00:00:09.42
points_100k 0 5717 00:00:05.71

_______________________________________2

lines_100k 0 27742 00:00:27.74
points_100k 0 29541 00:00:29.54
polygons_100k 0 30233 00:00:30.23
lines_100k 0 27833 00:00:27.83
points_100k 0 29770 00:00:29.77
polygons_100k 0 30280 00:00:30.28

_______________________________________3

lines_100k 0 28432 00:00:28.43
points_100k 0 30356 00:00:30.35
polygons_100k 0 31082 00:00:31.08
lines_100k 0 28599 00:00:28.59
points_100k 0 30901 00:00:30.90
polygons_100k 0 31326 00:00:31.32

_______________________________________4

lines_100k 0 29076 00:00:29.07
points_100k 0 30963 00:00:30.96
polygons_100k 0 31499 00:00:31.49
lines_100k 0 9108 00:00:09.10
points_100k 0 10008 00:00:10.00
polygons_100k 0 7356 00:00:07.35

_______________________________________5

lines_100k 0 27075 00:00:27.07
points_100k 0 29343 00:00:29.34
polygons_100k 0 29817 00:00:29.81
lines_100k 0 26662 00:00:26.66
points_100k 0 28816 00:00:28.81
polygons_100k 0 29322 00:00:29.32

_______________________________________6

lines_100k 0 27372 00:00:27.37
points_100k 0 29720 00:00:29.72
polygons_100k 0 30106 00:00:30.10
lines_100k 0 28676 00:00:28.67
points_100k 0 30680 00:00:30.68
polygons_100k 0 31249 00:00:31.24

_______________________________________7

lines_100k 0 29647 00:00:29.64
points_100k 0 31663 00:00:31.66
polygons_100k 0 32264 00:00:32.26
points_100k 0 22675 00:00:22.67
lines_100k 0 22180 00:00:22.18
polygons_100k 0 23649 00:00:23.64

_______________________________________8

lines_100k 0 31392 00:00:31.39
points_100k 0 33437 00:00:33.43
polygons_100k 0 33795 00:00:33.79
points_100k 0 9497 00:00:09.49
lines_100k 0 8767 00:00:08.76
polygons_100k 0 7251 00:00:07.25

9

lines_100k	0	28329	00:00:28.32
points_100k	0	30322	00:00:30.32
polygons_100k	0	30785	00:00:30.78
points_100k	0	5723	00:00:05.72
lines_100k	0	7953	00:00:07.95
polygons_100k	0	9610	00:00:09.61

10

lines_100k	0	26521	00:00:26.52
points_100k	0	28543	00:00:28.54
polygons_100k	0	29077	00:00:29.07
lines_100k	0	29859	00:00:29.85
points_100k	0	31892	00:00:31.89
polygons_100k	0	32480	00:00:32.48

TEST 3 ALL 12 LAYERS

1

points_100	0	1997	00:00:01.99
polygons_1k	0	4276	00:00:04.27
polygons_100	0	4379	00:00:04.37
points_1k	0	4434	00:00:04.43
lines_100	0	4479	00:00:04.47
lines_1k	0	4521	00:00:04.52
lines_10k	0	7954	00:00:07.95
polygons_10k	0	9240	00:00:09.24
points_10k	0	10700	00:00:10.70
lines_100k	0	30627	00:00:30.62
points_100k	0	32582	00:00:32.58
polygons_100k	0	33229	00:00:33.22

2

polygons_100	0	2155	00:00:02.15
lines_100	0	4473	00:00:04.47
points_100	0	4831	00:00:04.83
lines_1k	0	4843	00:00:04.84
polygons_1k	0	5103	00:00:05.10
points_1k	0	5353	00:00:05.35
points_10k	0	9910	00:00:09.91
lines_10k	0	7934	00:00:07.93
polygons_10k	0	8169	00:00:08.16
lines_100k	0	35180	00:00:35.18
points_100k	0	37355	00:00:37.35
polygons_100k	0	37154	00:00:37.15

3

polygons_100	0	2750	00:00:02.75
polygons_1k	0	4283	00:00:04.28
lines_10k	0	6622	00:00:06.62
points_1k	0	6559	00:00:06.55
points_10k	0	7240	00:00:07.24
points_100	0	4726	00:00:04.72
lines_100	0	2109	00:00:02.10
points_100k	0	12839	00:00:12.83
lines_1k	0	2597	00:00:02.59
polygons_10k	0	5969	00:00:05.96
lines_100k	0	13199	00:00:13.19
polygons_100k	0	14594	00:00:14.59

4

points_100	0	3713	00:00:03.71
lines_100	0	3986	00:00:03.98
polygons_100	0	4231	00:00:04.23
points_1k	0	5717	00:00:05.71
lines_1k	0	5300	00:00:05.30
polygons_1k	0	5773	00:00:05.77
lines_10k	0	7120	00:00:07.12
lines_100k	0	21722	00:00:21.72
points_10k	0	14695	00:00:14.69
polygons_10k	0	10157	00:00:10.15
polygons_100k	0	33140	00:00:29.14
points_100k	0	15219	00:00:15.21

5

polygons_100	0	2153	00:00:02.15
points_100	0	2830	00:00:02.83
lines_100	0	3111	00:00:03.11
lines_1k	0	3506	00:00:03.50
polygons_1k	0	3441	00:00:03.44
points_1k	0	3378	00:00:03.37
lines_10k	0	4115	00:00:04.15
polygons_10k	0	5322	00:00:05.32
points_10k	0	7180	00:00:07.18
lines_100k	0	33469	00:00:33.46
points_100k	0	35139	00:00:35.13
polygons_100k	0	35699	00:00:35.69

6

polygons_100	0	1685	00:00:01.68
lines_100	0	1935	00:00:01.93
points_100	0	2023	00:00:02.23
polygons_1k	0	2658	00:00:02.65
lines_1k	0	2648	00:00:02.64
points_1k	0	2990	00:00:02.99
points_10k	0	5392	00:00:05.39
polygons_10k	0	5959	00:00:05.95
lines_10k	0	6049	00:00:06.04
lines_100k	0	30224	00:00:30.22
points_100k	0	32137	00:00:32.13
polygons_100k	0	32362	00:00:32.36

7

lines_100	0	2304	00:00:02.30
points_100	0	2311	00:00:02.31
polygons_1k	0	3695	00:00:03.69
polygons_100	0	5500	00:00:05.50
lines_1k	0	6224	00:00:06.22
points_1k	0	6260	00:00:06.26
lines_10k	0	9858	00:00:09.85
points_10k	0	10798	00:00:10.79
polygons_10k	0	6526	00:00:06.52
lines_100k	0	34023	00:00:34.02
points_100k	0	36769	00:00:36.76
polygons_100k	0	36226	00:00:36.22

8

lines_100	0	1913	00:00:01.91
points_100	0	1972	00:00:01.97
polygons_1k	0	2235	00:00:02.23
polygons_100	0	2293	00:00:02.29
lines_1k	0	2322	00:00:02.32
points_1k	0	2386	00:00:02.38
lines_10k	0	6121	00:00:06.12
points_10k	0	7817	00:00:07.81
polygons_10k	0	9606	00:00:09.60
lines_100k	0	33100	00:00:33.10
points_100k	0	35127	00:00:35.12
polygons_100k	0	35222	00:00:35.22

9

lines_100	0	2302	00:00:02.30
points_100	0	2360	00:00:02.36
points_1k	0	2383	00:00:02.38
polygons_100	0	2400	00:00:02.40
lines_1k	0	2727	00:00:02.72
polygons_1k	0	2838	00:00:02.83
points_10k	0	3955	00:00:03.95
lines_10k	0	6052	00:00:06.05
polygons_10k	0	6975	00:00:06.97
lines_100k	0	32708	00:00:32.70
points_100k	0	34759	00:00:34.75
polygons_100k	0	34816	00:00:34.81

10

polygons_100	0	2400	00:00:02.40

points_100	0	1978	00:00:01.97
lines_100	0	2050	00:00:02.05
points_1k	0	2706	00:00:02.70
polygons_1k	0	2278	00:00:02.27
lines_1k	0	2145	00:00:02.14
lines_10k	0	5625	00:00:05.62
polygons_10k	0	6460	00:00:06.46
points_10k	0	7227	00:00:07.22
lines_100k	0	32060	00:00:32.06
points_100k	0	34150	00:00:34.15
polygons_100k	0	34038	00:00:34.03

____

________________________________9/12________________________________

____

lines_100	0	1099	00:00:01.09
lines_100	0	1094	00:00:01.09
lines_100	0	1099	00:00:01.09
lines_100	0	1097	00:00:01.09
lines_100	0	1099	00:00:01.09
lines_100	0	1096	00:00:01.09
lines_100	0	1095	00:00:01.09
lines_100	0	1101	00:00:01.10
lines_100	0	1094	00:00:01.09
lines_100	0	1100	00:00:01.10

____

points_100	0	1101	00:00:01.10
points_100	0	1098	00:00:01.09
points_100	0	1102	00:00:01.10
points_100	0	1101	00:00:01.10
points_100	0	1101	00:00:01.10
points_100	0	1113	00:00:01.11
points_100	0	1107	00:00:01.10
points_100	0	1106	00:00:01.10
points_100	0	1097	00:00:01.09
points_100	0	1111	00:00:01.11

____

points_100	0	1099	00:00:01.09
points_1k	0	1181	00:00:01.18
points_10k	0	1681	00:00:01.68
points_100k	0	5809	00:00:05.80
lines_100	0	1087	00:00:01.08
lines_1k	0	1192	00:00:01.19
lines_10k	0	1606	00:00:01.60
lines_100k	0	5553	00:00:05.55
polygons_100	0	1146	00:00:01.14
polygons_1k	0	1138	00:00:01.13
polygons_10k	0	1806	00:00:01.80
polygons_100k	0	6603	00:00:06.60

____________________points_1000__________SLICE TEST________________________________

__________________________points1000

points_1k	0	1184	00:00:01.18
points_1k	0	1170	00:00:01.17
points_1k	0	1175	00:00:01.17
points_1k	0	1381	00:00:01.38
points_1k	0	1169	00:00:01.16
points_1k	0	1172	00:00:01.17
points_1k	0	1169	00:00:01.16
points_1k	0	1173	00:00:01.17
points_1k	0	1162	00:00:01.16
points_1k	0	1172	00:00:01.17

____________________________points 2000

points_10k	0	1122	00:00:01.12
points_10k	0	1033	00:00:01.03
points_10k	0	1017	00:00:01.01
points_10k	0	1022	00:00:01.02
points_10k	0	1018	00:00:01.01
points_10k	0	1136	00:00:01.13
points_10k	0	1022	00:00:01.02
points_10k	0	1018	00:00:01.01
points_10k	0	1014	00:00:01.01
points_10k	0	1024	00:00:01.02

____________________________points3000

points_10k	0	1199	00:00:01.19
points_10k	0	1345	00:00:01.34
points_10k	0	1104	00:00:01.10
points_10k	0	1111	00:00:01.11
points_10k	0	1167	00:00:01.16
points_10k	0	1301	00:00:01.30
points_10k	0	1117	00:00:01.11
points_10k	0	1106	00:00:01.10
points_10k	0	1221	00:00:01.22
points_10k	0	1335	00:00:01.33

__________________________points 4000

points_10k	0	1187	00:00:01.18
points_10k	0	1264	00:00:01.26
points_10k	0	1174	00:00:01.17
points_10k	0	1384	00:00:01.38
points_10k	0	1280	00:00:01.28
points_10k	0	1178	00:00:01.17
points_10k	0	1260	00:00:01.26
points_10k	0	1363	00:00:01.36
points_10k	0	1174	00:00:01.17
points_10k	0	1229	00:00:01.22

____________________________points 5000

points_10k	0	1249	00:00:01.24
points_10k	0	1534	00:00:01.53
points_10k	0	1233	00:00:01.23
points_10k	0	1224	00:00:01.22
points_10k	0	1364	00:00:01.36
points_10k	0	1431	00:00:01.43
points_10k	0	1315	00:00:01.31
points_10k	0	1225	00:00:01.22
points_10k	0	1308	00:00:01.30
points_10k	0	1450	00:00:01.45

____________________________points 6000

points_10k	0	2125	00:00:02.12
points_10k	0	1375	00:00:01.37
points_10k	0	1376	00:00:01.37
points_10k	0	1494	00:00:01.49
points_10k	0	1381	00:00:01.38
points_10k	0	1488	00:00:01.48
points_10k	0	1599	00:00:01.59
points_10k	0	1393	00:00:01.39
points_10k	0	1294	00:00:01.29
points_10k	0	1565	00:00:01.56

____________________________points 7000

points_10k	0	2163	00:00:02.16
points_10k	0	1494	00:00:01.49
points_10k	0	1562	00:00:01.56
points_10k	0	1501	00:00:01.50
points_10k	0	1466	00:00:01.46
points_10k	0	1534	00:00:01.53
points_10k	0	1467	00:00:01.46
points_10k	0	1466	00:00:01.46
points_10k	0	1341	00:00:01.34
points_10k	0	1652	00:00:01.65

____________________________points 8000

points_10k	0	1494	00:00:01.49
points_10k	0	1707	00:00:01.70
points_10k	0	1391	00:00:01.39
points_10k	0	1476	00:00:01.47
points_10k	0	1670	00:00:01.67
points_10k	0	1495	00:00:01.49
points_10k	0	1378	00:00:01.37
points_10k	0	1700	00:00:01.70
points_10k	0	1465	00:00:01.46
points_10k	0	1500	00:00:01.50

points 9000

points_10k	0	1450	00:00:01.45
points_10k	0	1724	00:00:01.72
points_10k	0	1525	00:00:01.52
points_10k	0	1560	00:00:01.56
points_10k	0	1542	00:00:01.54
points_10k	0	1740	00:00:01.74
points_10k	0	1431	00:00:01.43
points_10k	0	1539	00:00:01.53
points_10k	0	1748	00:00:01.74
points_10k	0	1548	00:00:01.54

---

points 10k

points_10k	0	1682	00:00:01.68
points_10k	0	1873	00:00:01.87
points_10k	0	1639	00:00:01.63
points_10k	0	1646	00:00:01.64
points_10k	0	1644	00:00:01.64
points_10k	0	1842	00:00:01.84
points_10k	0	1652	00:00:01.65
points_10k	0	1643	00:00:01.64
points_10k	0	1644	00:00:01.64
points_10k	0	1650	00:00:01.65

points 20 000

points_100k	0	2150	00:00:02.15
points_100k	0	1919	00:00:01.91
points_100k	0	1919	00:00:01.91
points_100k	0	2297	00:00:02.29
points_100k	0	1935	00:00:01.93
points_100k	0	1950	00:00:01.95
points_100k	0	2130	00:00:02.13
points_100k	0	1935	00:00:01.93
points_100k	0	2121	00:00:02.12
points_10k	0	1183	00:00:01.18

points 30 000

points_100k	0	2577	00:00:02.57
points_100k	0	2333	00:00:02.33
points_100k	0	2609	00:00:02.60
points_100k	0	2562	00:00:02.56
points_100k	0	2374	00:00:02.37
points_100k	0	2634	00:00:02.63
points_100k	0	2653	00:00:02.65
points_100k	0	2356	00:00:02.35
points_100k	0	2363	00:00:02.36
points_100k	0	2750	00:00:02.75

points 40 000

points_100k	0	2788	00:00:02.78
points_100k	0	2961	00:00:02.96
points_100k	0	3074	00:00:03.07
points_100k	0	2776	00:00:02.77
points_100k	0	3076	00:00:03.07
points_100k	0	3004	00:00:03.00
points_100k	0	2861	00:00:02.86
points_100k	0	2807	00:00:02.80
points_100k	0	2981	00:00:02.98
points_100k	0	3108	00:00:03.10

points 50 000

points_100k	0	3277	00:00:03.27
points_100k	0	3773	00:00:03.77
points_100k	0	3252	00:00:03.25
points_100k	0	3518	00:00:03.51
points_100k	0	3439	00:00:03.43
points_100k	0	3487	00:00:03.48
points_100k	0	3228	00:00:03.22
points_100k	0	3685	00:00:03.68
points_100k	0	3240	00:00:03.24
points_100k	0	3463	00:00:03.46

points 60 000

points_100k	0	4555	00:00:04.55
points_100k	0	4033	00:00:04.03
points_100k	0	3651	00:00:03.65
points_100k	0	4098	00:00:04.09
points_100k	0	3921	00:00:03.92
points_100k	0	3678	00:00:03.67
points_100k	0	4149	00:00:04.14
points_100k	0	3643	00:00:03.64
points_100k	0	3909	00:00:03.90
points_100k	0	4133	00:00:04.13

points 70000

points_100k	0	4137	00:00:04.13
points_100k	0	4596	00:00:04.59
points_100k	0	4379	00:00:04.37
points_100k	0	4168	00:00:04.16
points_100k	0	4506	00:00:04.50
points_100k	0	4355	00:00:04.35
points_100k	0	4068	00:00:04.06
points_100k	0	4354	00:00:04.35
points_100k	0	4365	00:00:04.36
points_100k	0	4302	00:00:04.30

points 80 000

points_100k	0	4694	00:00:04.69
points_100k	0	4746	00:00:04.74
points_100k	0	4973	00:00:04.97
points_100k	0	4732	00:00:04.73
points_100k	0	4849	00:00:04.84
points_100k	0	4549	00:00:04.54
points_100k	0	4945	00:00:04.94
points_100k	0	4751	00:00:04.75
points_100k	0	4957	00:00:04.95
points_100k	0	4563	00:00:04.56

points 90 000

points_100k	0	5180	00:00:05.18
points_100k	0	5060	00:00:05.06
points_100k	0	5277	00:00:05.27
points_100k	0	5060	00:00:05.06
points_100k	0	5303	00:00:05.30
points_100k	0	5093	00:00:05.09
points_100k	0	5268	00:00:05.26
points_100k	0	5076	00:00:05.07
points_100k	0	5214	00:00:05.21
points_100k	0	4932	00:00:04.93

POLYGONS

20 000

polygons_100k	0	2066	00:00:02.06
polygons_100k	0	2187	00:00:02.18
polygons_100k	0	1986	00:00:01.98
polygons_100k	0	1991	00:00:01.99
polygons_100k	0	1987	00:00:01.98
polygons_100k	0	2258	00:00:02.25
polygons_100k	0	1983	00:00:01.98

polygons_100k	0	1992	00:00:01.99
polygons_100k	0	1984	00:00:01.98
polygons_100k	0	2173	00:00:02.17

polygons 30 000

polygons_100k	0	2664	00:00:02.66
polygons_100k	0	2534	00:00:02.53
polygons_100k	0	2772	00:00:02.77
polygons_100k	0	2647	00:00:02.64
polygons_100k	0	2732	00:00:02.73
polygons_100k	0	2546	00:00:02.54
polygons_100k	0	2590	00:00:02.59
polygons_100k	0	2767	00:00:02.76
polygons_100k	0	2531	00:00:02.53
polygons_100k	0	2549	00:00:02.54

polygons 40 000

polygons_100k	0	2859	00:00:02.85
polygons_100k	0	2512	00:00:02.51
polygons_100k	0	2530	00:00:02.53
polygons_100k	0	2816	00:00:02.81
polygons_100k	0	2525	00:00:02.52
polygons_100k	0	2532	00:00:02.53
polygons_100k	0	2787	00:00:02.78
polygons_100k	0	2561	00:00:02.56
polygons_100k	0	2530	00:00:02.53
polygons_100k	0	2754	00:00:02.75

polygons 50 000

polygons_100k	0	3786	00:00:03.78
polygons_100k	0	3958	00:00:03.95
polygons_100k	0	3749	00:00:03.74
polygons_100k	0	3905	00:00:03.90
polygons_100k	0	3756	00:00:03.75
polygons_100k	0	3902	00:00:03.90
polygons_100k	0	3746	00:00:03.74
polygons_100k	0	3891	00:00:03.89
polygons_100k	0	3736	00:00:03.73
polygons_100k	0	3900	00:00:03.90

polygons 60 000

polygons_100k	0	4341	00:00:04.34
polygons_100k	0	4534	00:00:04.53
polygons_100k	0	4257	00:00:04.25
polygons_100k	0	4544	00:00:04.54
polygons_100k	0	4217	00:00:04.21
polygons_100k	0	4537	00:00:04.53
polygons_100k	0	4305	00:00:04.30
polygons_100k	0	4480	00:00:04.48
polygons_100k	0	4319	00:00:04.31
polygons_100k	0	4436	00:00:04.43

polygons 70 000

polygons_100k	0	4885	00:00:04.88
polygons_100k	0	5043	00:00:05.04
polygons_100k	0	4675	00:00:04.67
polygons_100k	0	5037	00:00:05.03
polygons_100k	0	4883	00:00:04.88
polygons_100k	0	5040	00:00:05.04
polygons_100k	0	5068	00:00:05.06
polygons_100k	0	4891	00:00:04.89
polygons_100k	0	5044	00:00:05.04
polygons_100k	0	4816	00:00:04.81

polygons 80 000

polygons_100k	0	5588	00:00:05.58
polygons_100k	0	5397	00:00:05.39
polygons_100k	0	5603	00:00:05.60
polygons_100k	0	5310	00:00:05.31
polygons_100k	0	5573	00:00:05.57
polygons_100k	0	5588	00:00:05.58
polygons_100k	0	5603	00:00:05.60
polygons_100k	0	5412	00:00:05.41
polygons_100k	0	5541	00:00:05.54
polygons_100k	0	5600	00:00:05.60

polygons 90 000

polygons_100k	0	6216	00:00:06.21
polygons_100k	0	6157	00:00:06.15
polygons_100k	0	6139	00:00:06.13
polygons_100k	0	6209	00:00:06.20
polygons_100k	0	6172	00:00:06.17
polygons_100k	0	6098	00:00:06.09
polygons_100k	0	6127	00:00:06.12
polygons_100k	0	6180	00:00:06.18
polygons_100k	0	6205	00:00:06.20
polygons_100k	0	6186	00:00:06.18

TIME WATCH

LAYER			SQLEXPRESS		GEOSERVER	SERVICE.ASHX	
points_100	0	0	8	0	1031	1040	00:00:01.04
points_100	0	0	7	0	1037	1045	00:00:01.04
points_100	0	0	7	0	1031	1039	00:00:01.03
points_100	0	0	7	0	1044	1051	00:00:01.05
points_100	0	0	8	0	1035	1044	00:00:01.04
points_100	0	0	7	0	1046	1054	00:00:01.05
points_100	0	0	7	0	1035	1043	00:00:01.04
points_100	0	0	7	0	1036	1044	00:00:01.04
points_100	0	0	7	0	1036	1044	00:00:01.04
points_100	0	0	7	0	1045	1053	00:00:01.05
points_100	0	0	8	0	1029	1038	00:00:01.03
points_100	0	0	7	0	1039	1047	00:00:01.04
points_100	0	0	7	0	1032	1039	00:00:01.03
points_100	0	0	8	0	1035	1043	00:00:01.04
points_100	0	0	7	0	1035	1042	00:00:01.04
points_100	0	0	11	0	1045	1057	00:00:01.05
points_100	0	0	8	0	1040	1049	00:00:01.04
points_100	0	0	7	0	1048	1055	00:00:01.05
points_100	0	0	8	0	1026	1035	00:00:01.03
points_100	0	0	7	0	1031	1038	00:00:01.03
points_1k	0	0	8	0	1117	1126	00:00:01.12
points_1k	0	0	7	0	1121	1129	00:00:01.12

points_1k	0	0	8	0	1114	1122	00:00:01.12
points_1k	0	0	7	0	1129	1136	00:00:01.13
points_1k	0	0	7	0	1117	1124	00:00:01.12
points_1k	0	0	7	0	1129	1137	00:00:01.13
points_1k	0	0	7	0	1115	1123	00:00:01.12
points_1k	0	0	7	0	1127	1135	00:00:01.13
points_1k	0	0	7	0	1118	1126	00:00:01.12
points_1k	0	0	7	0	1126	1134	00:00:01.13
points_1k	0	0	8	0	1108	1117	00:00:01.11
points_1k	0	0	7	0	1124	1131	00:00:01.13
points_1k	0	0	7	0	1113	1120	00:00:01.12
points_1k	0	0	7	0	1118	1126	00:00:01.12
points_1k	0	0	7	0	1113	1120	00:00:01.12
points_1k	0	0	7	0	1123	1130	00:00:01.13
points_1k	0	0	7	0	1109	1117	00:00:01.11
points_1k	0	0	6	0	1118	1125	00:00:01.12
points_1k	0	0	7	0	1109	1118	00:00:01.11
points_1k	0	0	7	0	1130	1137	00:00:01.13
points_10k	0	0	9	0	1606	1615	00:00:01.61
points_10k	0	0	7	0	1615	1623	00:00:01.62
points_10k	0	0	7	0	1597	1605	00:00:01.60
points_10k	0	0	7	0	1819	1826	00:00:01.82
points_10k	0	0	7	0	1602	1610	00:00:01.61
points_10k	0	0	7	0	1613	1621	00:00:01.62
points_10k	0	0	7	0	1603	1611	00:00:01.61
points_10k	0	0	7	0	1615	1623	00:00:01.62
points_10k	0	0	7	0	1780	1787	00:00:01.78
points_10k	0	0	8	0	1639	1647	00:00:01.64
points_10k	0	0	8	0	1597	1605	00:00:01.60
points_10k	0	0	7	0	1612	1619	00:00:01.61
points_10k	0	0	7	0	1775	1783	00:00:01.78
points_10k	0	0	7	0	1610	1617	00:00:01.61
points_10k	0	0	7	0	1603	1610	00:00:01.61
points_10k	0	0	7	0	1603	1611	00:00:01.61
points_10k	0	0	7	0	1778	1785	00:00:01.78
points_10k	0	0	7	0	1610	1618	00:00:01.61
points_10k	0	0	7	0	1603	1611	00:00:01.61
points_10k	0	0	7	0	1605	1612	00:00:01.61
points_100k	0	0	8	0	5641	5650	00:00:05.65
points_100k	0	0	7	0	5455	5463	00:00:05.46
points_100k	0	0	7	0	5642	5649	00:00:05.64
points_100k	0	0	7	0	5411	5419	00:00:05.41
points_100k	0	0	8	0	5614	5623	00:00:05.62
points_100k	0	0	7	0	5443	5451	00:00:05.45
points_100k	0	0	8	0	5602	5610	00:00:05.61
points_100k	0	0	7	0	5404	5412	00:00:05.41
points_100k	0	0	7	0	5599	5607	00:00:05.60
points_100k	0	0	7	0	5417	5424	00:00:05.42
points_100k	0	0	8	0	5602	5611	00:00:05.61
points_100k	0	0	8	0	5398	5407	00:00:05.40
points_100k	0	0	9	0	5577	5586	00:00:05.58
points_100k	0	0	7	0	5382	5390	00:00:05.39
points_100k	0	0	7	0	5570	5578	00:00:05.57
points_100k	0	0	7	0	5405	5412	00:00:05.41
points_100k	0	0	7	0	5613	5621	00:00:05.62
points_100k	0	0	8	0	5427	5435	00:00:05.43
points_100k	0	0	8	0	5587	5595	00:00:05.59
points_100k	0	0	7	0	5427	5435	00:00:05.43
lines_100	0	0	7	0	1078	1086	00:00:01.08
lines_100	0	0	7	0	1019	1027	00:00:01.02
lines_100	0	0	7	0	1014	1022	00:00:01.02
lines_100	0	0	7	0	1019	1027	00:00:01.02
lines_100	0	0	7	0	1013	1021	00:00:01.02
lines_100	0	0	8	0	1021	1030	00:00:01.03
lines_100	0	0	8	0	1016	1024	00:00:01.02

lines_100	0	0	7	0	1014	1021	00:00:01.02
lines_100	0	0	7	0	1013	1021	00:00:01.02
lines_100	0	0	7	0	1023	1031	00:00:01.03
lines_100	0	0	7	0	1018	1026	00:00:01.02
lines_100	0	0	7	0	1017	1025	00:00:01.02
lines_100	0	0	7	0	1011	1019	00:00:01.01
lines_100	0	0	7	0	1025	1033	00:00:01.03
lines_100	0	0	7	0	1007	1015	00:00:01.01
lines_100	0	0	7	0	1022	1029	00:00:01.02
lines_100	0	0	7	0	1016	1024	00:00:01.02
lines_100	0	0	7	0	1019	1026	00:00:01.02
lines_100	0	0	9	0	1011	1020	00:00:01.02
lines_100	0	0	8	0	1033	1042	00:00:01.04
lines_1k	0	0	7	0	1116	1124	00:00:01.12
lines_1k	0	0	7	0	1113	1121	00:00:01.12
lines_1k	0	0	7	0	1111	1118	00:00:01.11
lines_1k	0	0	7	0	1125	1132	00:00:01.13
lines_1k	0	0	7	0	1115	1122	00:00:01.12
lines_1k	0	0	7	0	1124	1131	00:00:01.13
lines_1k	0	0	9	0	1114	1123	00:00:01.12
lines_1k	0	0	8	0	1119	1127	00:00:01.12
lines_1k	0	0	7	0	1109	1117	00:00:01.11
lines_1k	0	0	8	0	1142	1151	00:00:01.15
lines_1k	0	0	7	0	1115	1122	00:00:01.12
lines_1k	0	0	7	0	1128	1136	00:00:01.13
lines_1k	0	0	8	0	1118	1126	00:00:01.12
lines_1k	0	0	7	0	1119	1127	00:00:01.12
lines_1k	0	0	7	0	1111	1118	00:00:01.11
lines_1k	0	0	7	0	1118	1126	00:00:01.12
lines_1k	0	0	7	0	1111	1119	00:00:01.11
lines_1k	0	0	7	0	1118	1126	00:00:01.12
lines_1k	0	0	7	0	1112	1120	00:00:01.12
lines_1k	0	0	7	0	1121	1129	00:00:01.12
lines_10k	0	0	8	0	1536	1545	00:00:01.54
lines_10k	0	0	7	0	1551	1558	00:00:01.55
lines_10k	0	0	7	0	1730	1738	00:00:01.73
lines_10k	0	0	7	0	1559	1567	00:00:01.56
lines_10k	0	0	7	0	1535	1543	00:00:01.54
lines_10k	0	0	7	0	1559	1566	00:00:01.56
lines_10k	0	0	7	0	1545	1552	00:00:01.55
lines_10k	0	0	7	0	1740	1747	00:00:01.74
lines_10k	0	0	7	0	1540	1548	00:00:01.54
lines_10k	0	0	7	0	1549	1557	00:00:01.55
lines_10k	0	0	8	0	1546	1555	00:00:01.55
lines_10k	0	0	7	0	1552	1560	00:00:01.56
lines_10k	0	0	7	0	1771	1778	00:00:01.77
lines_10k	0	0	8	0	1553	1561	00:00:01.56
lines_10k	0	0	7	0	1540	1548	00:00:01.54
lines_10k	0	0	7	0	1549	1556	00:00:01.55
lines_10k	0	0	7	0	1715	1722	00:00:01.72
lines_10k	0	0	7	0	1551	1558	00:00:01.55
lines_10k	0	0	7	0	1538	1545	00:00:01.54
lines_10k	0	0	7	0	1548	1556	00:00:01.55
lines_100k	0	0	8	0	5512	5521	00:00:05.52
lines_100k	0	0	8	0	5542	5551	00:00:05.55
lines_100k	0	0	7	0	5504	5512	00:00:05.51
lines_100k	0	0	7	0	5523	5531	00:00:05.53
lines_100k	0	0	8	0	5524	5533	00:00:05.53
lines_100k	0	0	7	0	5562	5569	00:00:05.56
lines_100k	0	0	7	0	5510	5518	00:00:05.51
lines_100k	0	0	7	0	5574	5582	00:00:05.58
lines_100k	0	0	8	0	5516	5525	00:00:05.52
lines_100k	0	0	7	0	5466	5473	00:00:05.47
lines_100k	0	0	7	0	5460	5467	00:00:05.46
lines_100k	0	0	8	0	5471	5480	00:00:05.48
lines_100k	0	0	8	0	5490	5499	00:00:05.49
lines_100k	0	0	8	0	5713	5722	00:00:05.72
lines_100k	0	0	7	0	5682	5690	00:00:05.69

lines_100k	0	0	7	0	5485	5493	00:00:05.49
lines_100k	0	0	8	0	5721	5730	00:00:05.73
lines_100k	0	0	7	0	5482	5490	00:00:05.49
lines_100k	0	0	8	0	5707	5715	00:00:05.71
lines_100k	0	0	7	0	5897	5905	00:00:05.90
lines_100k	0	0	7	0	5470	5478	00:00:05.47

polygons_100	0	0	8	0	1083	1092	00:00:01.09
polygons_100	0	0	7	0	1012	1020	00:00:01.02
polygons_100	0	0	7	0	1002	1010	00:00:01.01
polygons_100	0	0	7	0	1005	1013	00:00:01.01
polygons_100	0	0	7	0	1002	1009	00:00:01.00
polygons_100	0	0	7	0	1011	1019	00:00:01.01
polygons_100	0	0	8	0	1007	1016	00:00:01.01
polygons_100	0	0	8	0	1005	1014	00:00:01.01
polygons_100	0	0	7	0	1000	1007	00:00:01.00
polygons_100	0	0	7	0	1004	1011	00:00:01.01
polygons_100	0	0	7	0	1000	1008	00:00:01.00
polygons_100	0	0	7	0	1001	1009	00:00:01.00
polygons_100	0	0	7	0	1001	1009	00:00:01.00
polygons_100	0	0	7	0	1004	1012	00:00:01.01
polygons_100	0	0	7	0	1002	1010	00:00:01.01
polygons_100	0	0	7	0	1003	1011	00:00:01.01
polygons_100	0	0	7	0	1002	1009	00:00:01.00
polygons_100	0	0	7	0	1004	1012	00:00:01.01
polygons_100	0	0	7	0	1002	1009	00:00:01.00
polygons_100	0	0	7	0	1004	1013	00:00:01.01

polygons_1k	0	0	8	0	1076	1084	00:00:01.08
polygons_1k	0	0	7	0	1072	1080	00:00:01.08
polygons_1k	0	0	7	0	1057	1065	00:00:01.06
polygons_1k	0	0	7	0	1063	1070	00:00:01.07
polygons_1k	0	0	6	0	1055	1061	00:00:01.06
polygons_1k	0	0	7	0	1060	1068	00:00:01.06
polygons_1k	0	0	7	0	1058	1066	00:00:01.06
polygons_1k	0	0	7	0	1061	1069	00:00:01.06
polygons_1k	0	0	8	0	1055	1064	00:00:01.06
polygons_1k	0	0	7	0	1063	1071	00:00:01.07
polygons_1k	0	0	7	0	1062	1070	00:00:01.07
polygons_1k	0	0	7	0	1060	1068	00:00:01.06
polygons_1k	0	0	7	0	1056	1063	00:00:01.06
polygons_1k	0	0	7	0	1058	1065	00:00:01.06
polygons_1k	0	0	7	0	1061	1068	00:00:01.06
polygons_1k	0	0	7	0	1071	1078	00:00:01.07
polygons_1k	0	0	8	0	1064	1073	00:00:01.07
polygons_1k	0	0	8	0	1066	1075	00:00:01.07
polygons_1k	0	0	8	0	1060	1069	00:00:01.06
polygons_1k	0	0	7	0	1066	1073	00:00:01.07

polygons_10k	0	0	7	0	1535	1542	00:00:01.54
polygons_10k	0	0	7	0	1540	1548	00:00:01.54
polygons_10k	0	0	8	0	1536	1544	00:00:01.54
polygons_10k	0	0	7	0	1724	1732	00:00:01.73
polygons_10k	0	0	8	0	1533	1541	00:00:01.54
polygons_10k	0	0	7	0	1532	1539	00:00:01.53
polygons_10k	0	0	7	0	1540	1548	00:00:01.54
polygons_10k	0	0	7	0	1546	1553	00:00:01.55
polygons_10k	0	0	7	0	1716	1724	00:00:01.72
polygons_10k	0	0	7	0	1535	1543	00:00:01.54
polygons_10k	0	0	7	0	1542	1549	00:00:01.54
polygons_10k	0	0	7	0	1540	1547	00:00:01.54
polygons_10k	0	0	7	0	1715	1723	00:00:01.72
polygons_10k	0	0	7	0	1567	1575	00:00:01.57
polygons_10k	0	0	7	0	1546	1553	00:00:01.55
polygons_10k	0	0	7	0	1568	1576	00:00:01.57
polygons_10k	0	0	8	0	1744	1753	00:00:01.75
polygons_10k	0	0	8	0	1584	1592	00:00:01.59
polygons_10k	0	0	7	0	1540	1548	00:00:01.54
polygons_10k	0	0	7	0	1561	1568	00:00:01.56

polygons_100k	0	0	9	0	6747	6756	00:00:06.75

polygons_100k	0	0	7	0	6790	6798	00:00:06.79
polygons_100k	0	0	8	0	6780	6789	00:00:06.78
polygons_100k	0	0	7	0	6783	6791	00:00:06.79
polygons_100k	0	0	7	0	6772	6779	00:00:06.77
polygons_100k	0	0	7	0	6781	6789	00:00:06.78
polygons_100k	0	0	7	0	6782	6790	00:00:06.79
polygons_100k	0	0	8	0	6758	6767	00:00:06.76
polygons_100k	0	0	7	0	6773	6781	00:00:06.78
polygons_100k	0	0	7	0	6806	6814	00:00:06.81
polygons_100k	0	0	7	0	6782	6790	00:00:06.79
polygons_100k	0	0	7	0	6767	6775	00:00:06.77
polygons_100k	0	0	8	0	6768	6776	00:00:06.77
polygons_100k	0	0	7	0	6759	6767	00:00:06.76
polygons_100k	0	0	7	0	6757	6765	00:00:06.76
polygons_100k	0	0	7	0	6788	6796	00:00:06.79
polygons_100k	0	0	8	0	6763	6772	00:00:06.77
polygons_100k	0	0	7	0	6780	6788	00:00:06.78
polygons_100k	0	0	7	0	6781	6789	00:00:06.78
polygons_100k	0	0	7	0	6772	6780	00:00:06.78

____10 december 2013_____________________________

______________________________controltest 100-1000 features______________________________________

points_100	0	0	8	0	1104	1113	00:00:01.11
points_100	0	0	6	0	1100	1107	00:00:01.10
points_100	0	0	7	0	1113	1121	00:00:01.12
points_100	0	0	6	0	1314	1321	00:00:01.32
points_100	0	0	6	0	1097	1104	00:00:01.10
points_100	0	0	6	0	1099	1105	00:00:01.10
points_100	0	0	6	0	1101	1108	00:00:01.10
points_100	0	0	6	0	1277	1283	00:00:01.28
points_100	0	0	6	0	1101	1108	00:00:01.10
points_100	0	0	6	0	1101	1108	00:00:01.10
points_100	0	0	6	0	1099	1106	00:00:01.10
points_100	0	0	7	0	1275	1282	00:00:01.28
points_100	0	0	7	0	1100	1108	00:00:01.10
points_100	0	0	6	0	1096	1102	00:00:01.10
points_100	0	0	6	0	1105	1112	00:00:01.11
points_100	0	0	6	0	1278	1285	00:00:01.28
points_100	0	0	7	0	1107	1115	00:00:01.11
points_100	0	0	6	0	1097	1104	00:00:01.10
points_100	0	0	6	0	1095	1102	00:00:01.10
points_100	0	0	6	0	1281	1288	00:00:01.28

lines_100	0	0	9	0	1160	1169	00:00:01.16
lines_100	0	0	6	0	1091	1098	00:00:01.09
lines_100	0	0	6	0	1094	1100	00:00:01.10
lines_100	0	0	6	0	1279	1286	00:00:01.28
lines_100	0	0	6	0	1095	1102	00:00:01.10
lines_100	0	0	6	0	1100	1107	00:00:01.10
lines_100	0	0	6	0	1094	1101	00:00:01.10
lines_100	0	0	6	0	1280	1287	00:00:01.28
lines_100	0	0	6	0	1094	1101	00:00:01.10
lines_100	0	0	7	0	1089	1097	00:00:01.09
lines_100	0	0	6	0	1089	1095	00:00:01.09
lines_100	0	0	6	0	1276	1283	00:00:01.28
lines_100	0	0	7	0	1091	1099	00:00:01.09
lines_100	0	0	6	0	1095	1102	00:00:01.10
lines_100	0	0	7	0	1096	1103	00:00:01.10
lines_100	0	0	6	0	1301	1307	00:00:01.30
lines_100	0	0	6	0	1093	1100	00:00:01.10
lines_100	0	0	6	0	1097	1104	00:00:01.10
lines_100	0	0	6	0	1087	1094	00:00:01.09
lines_100	0	0	6	0	1272	1279	00:00:01.27

polygons_100	0	0	9	0	1150	1159	00:00:01.15
polygons_100	0	0	6	0	1082	1088	00:00:01.08
polygons_100	0	0	6	0	1089	1096	00:00:01.09
polygons_100	0	0	6	0	1269	1276	00:00:01.27
polygons_100	0	0	7	0	1084	1091	00:00:01.09
polygons_100	0	0	7	0	1082	1089	00:00:01.08

polygons_100	0	0	6	0	1087	1093	00:00:01.09
polygons_100	0	0	7	0	1271	1278	00:00:01.27
polygons_100	0	0	7	0	1090	1098	00:00:01.09
polygons_100	0	0	7	0	1086	1093	00:00:01.09
polygons_100	0	0	6	0	1083	1090	00:00:01.09
polygons_100	0	0	6	0	1293	1300	00:00:01.30
polygons_100	0	0	6	0	1083	1090	00:00:01.09
polygons_100	0	0	6	0	1080	1087	00:00:01.08
polygons_100	0	0	6	0	1083	1090	00:00:01.09
polygons_100	0	0	6	0	1274	1281	00:00:01.28
polygons_100	0	0	6	0	1086	1093	00:00:01.09
polygons_100	0	0	6	0	1083	1090	00:00:01.09
polygons_100	0	0	6	0	1080	1087	00:00:01.08
polygons_100	0	0	6	0	1268	1275	00:00:01.27