Министерство природных ресурсов и экологии РФ

Федеральная служба по гидрометеорологии и мониторингу окружающей среды (Росгидромет)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ

«СИБИРСКИЙ РЕГИОНАЛЬНЫЙ НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ

ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ ИНСТИТУТ»

(ФГБУ «СИБНИГМИ»)

УДК 551.5:001.891.57
№ госрегистрации 01201180382
Инв. №

УТВЕРЖДАЮ

Директор _____

Крупчатников В.Н.

«_31_» _декабря_ 2013 г.

ОТЧЕТ

О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

Развитие систем визуализации фактической и прогностической метеорологической информации

(заключительный)

Шифр темы: 1.1.1.4

Научный руководитель

к.т.н. Колкер А.Б.

Новосибирск 2013

СПИСОК ИСПОЛНИТЕЛЕЙ

Руководитель темы,

к.т.н. _____А.Б.Колкер

Исполнители темы

_____А.В.Гочаков

_____М.С.Котов

_____В.М.Токарев

_____ Т.С. Гаврилова

Нормоконтролер _____Т.П.Панькова

РЕФЕРАТ

Отчет 113 с., 32 рис., 3 табл., 16 источников,

ВЕБ-ГИС ТЕХНОЛОГИИ, ВИЗУАЛИЗАЦИЯ МЕТЕОРОЛОГИЧЕСКИХ ДАННЫХ, ПОСТПРОЦЕССИНГ ДАННЫХ МЕЗОМАСШТАБНОГО МОДЕЛИРОВАНИЯ

Данная работа посвящена созданию картографического сервиса с использованием последних разработок в области web-картографии путем организации выделенного сервера или пула серверов. Картографический сервис представляет собой комплексную информационную систему, позволяющую визуализировать пространственную информацию, включая возможности масштабирования изображений, а также усваивать и обрабатывать данные ряда научных форматов. Разработка базируется на свободном программном обеспечении, предусмотрены возможности для её дальнейшего развития.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная работа посвящена задаче создания картографического сервиса, предназначенного для визуализации гео-ориентированных данных (в первую очередь метеорологических). Разрабатываемый сервис должен предоставлять возможность удобной визуализации данных, представленных в различных масштабах, а также способах хранения (в том числе сеточных данных высокого разрешения)

Специализированные картографические сервисы, созданные для использования узким кругом специалистов, появились довольно давно. В отечественной метеорологии в первую очередь следует отметить сервис ЕСИМО. Портал ЕСИМО посвящен глобальной системе визуализации данных на морях и океанах и по праву является пионером широкомасштабного использования таких технологий в области гидрометеорологии Другой тенденцией последних лет является появление глобальных картографических сервисов, таких как Google Maps. Эти сервисы характеризуются широкой географией, массовым распространением и быстротой доставки пространственных данных пользователю, что, несомненно, оказывает огромное влияние на отрасль web-картографии в целом, и задает новые стандарты. Особенностью данного решения является то, что вся основная картографическая инфраструктура базируется на сервисах и серверах компании Google, что подчас неприемлемо при организации сервисов в отраслевых сетях. Помимо прочего, работа с Google Maps возможна лишь в единственной проекции, что не всегда удобно для профессионального использования.

Данная работа посвящена созданию картографического сервиса с использованием последних разработок в области web-картографии путем организации выделенного сервера или пула серверов. Картографический сервис представляет собой комплексную информационную систему, позволяющую визуализировать пространственную информацию, а также обрабатывать данные ряда научных форматов. Разработка базируется на свободном программном обеспечении, с учетом возможности масштабирования изображений и возможности дальнейшего развития. В данный отчет вошли материалы магистерской диссертации «Разработка комплексной системы визуализации геофизических данных» Котова М.С., который является одним из ключевых исполнителей названной темы.

Постановка задачи

В данной работе требуется исследовать эффективность существующих решений и разработать комплексную информационную систему обработки и визуализации геофизических данных. Информационная система должна обеспечивать:

- Работу с наиболее популярными web-браузерами, без использования стороннего программного обеспечения;
- Поддержку векторных и растровых источников пространственных данных;
- Работу с медленными каналами связи;
- Возможность интеграции с СУБД;
- Масштабируемость;
- Работу в нескольких картографических проекциях (EPSG:4326, EPSG:3576);
- Возможность дальнейшего развития;
- Предоставление высококачественного картографического результата;
- Возможность оперировать с событиями, порождаемыми действиями пользователя;
- Поддержку одного из стандартов Open Geospatial Consortium.

Разработка информационной системы должна быть основана на использовании программных продуктов с открытым исходным кодом.

1 Обзор существующих решений

1.1 Определения

Для успешного выполнения обзора существующих решений рассматриваемой задачи или ее модификаций следует определить формулировку задачи в рамках модели предметной области. Так, считается, что комплексная информационная система - это совокупность технического, программного и организационного обеспечения, а также персонал, использующий эту систему. В рамках поставленной задачи и обозначенной предметной области оговорены требования к информационной системе с точки зрения именно программной части. Далее будет использоваться именно этот контекст.

Можно сказать, что разрабатываемая комплексная информационная система будет являться географической информационной (геоинформационной или ГИС) системой. Геоинформационная система – информационная система, предназначенная для сбора, хранения, анализа и графической визуализации пространственных данных и связанной с ними информации о представленных в ГИС объектах. Данный термин используется и в более узком смысле – ГИС как инструмент (программный продукт), позволяющий визуализировать, создавать и обрабатывать информацию об объектах цифровой карты. Существует большое количество типов ГИС, которые различаются географическим охватом, предметной областью и другими характеристиками. Но на сегодняшний день наибольший интерес представляют web-ориентированные геоинформационные системы, которые предоставляют картографический сервис и технологии для работы пользователей с пространственными данными.

В последние годы развитие web-технологий все в большей степени предоставляет пользователям возможность использовать новые средства доставки информации. Web-картография является областью компьютерных технологий, связанной с доставкой пространственных данных конечному пользователю. Приставка “*web*” используется для удобства, в качестве среды могут использоваться любые сети, интернет. Безусловно, web-картография является одним из направлений геоинформационных технологий в целом. Прямого аналога устоявшемуся термину “web-гис” в англоязычных источниках найти не удастся и гораздо чаще встречается такой термин как *web mapping service* (картографический web-сервис). Далее, упоминая термин “данные”, мы будем иметь в виду *пространственные данные*, то есть, данные

включающие координатную составляющую, привязывающую их к определенному месту. Основными задачами web-картографии являются:

- Визуализация существующей информации – пространственное представление информации.
- Облегчение работы с пространственной информацией в web, поиск, прокладка маршрутов и другие услуги, основанные на местоположении объектов.

1.2 Классификация web-картографических приложений

Разнообразие современных механизмов для создания web-картографических приложений велико и их классификация довольно затруднительна. Это происходит вследствие того, что каждый разработчик стремится создать как можно более комплексное средство, включающее технологии создания, визуализации и публикации данных. Тем не менее, можно выделить несколько основных видов приложений, имеющих отношение к web-картографии:

- *Web-глобусы* (Google Maps, Google Earth, Virtual Earth, ArcGIS Explorer и др.) – простое и эффективное средство быстрого создания и публикации данных в интернет. Данная категория инструментов, характеризуется массовым распространением и быстрой доставкой данных пользователю. Может использовать в качестве клиента как web-браузер, так и отдельное приложение. Как правило, включает доступ по умолчанию к некой картографической основе (подложке) – базе данных, что является одновременно их большим плюсом и не меньшим минусом, так как сменить эту основу в большинстве случаев нельзя. Также, как правило, этим инструментам свойственны проблемы при работе с большими объемами пользовательских данных, настраиваемостью, элементарным анализом (обрезка, пересечение слоев данных).
- *Пользовательские ГИС* (ArcGIS, Mapinfo, QGIS, gvSIG и др.) – большая и сложная категория, тесно связанная с web-картографией. Как правило, пользовательские ГИС, с одной стороны, играют роль клиентов, работающих с данными, поставляемыми картографическими web-серверами, и с другой - в них осуществляется массовая подготовка и анализ данных перед публикацией их в web.
- *Картографические web-сервера* (MapServer, GeoServer, OpenLayers и др.) – целое семейство продуктов свободного и проприетарного характера, предназначенных для быстрой публикации пользовательских данных в web. Эти инструменты позволяют создать

интерфейс нужной сложности, интегрировать сервис с базой данных, поддерживающей классы пространственных данных (PostgreSQL, SQL Server, MySQL, ArcSDE). Главным отличием подобных систем от таких сервисов, как Google Maps, является полный контроль над программным обеспечением и самими данными. Однако взамен приходится расплачиваться большой сложностью установки и настройки, часто требующей хотя бы начальных знаний языков программирования и основ администрирования.

1.3 Классификация представителей рынка web-ГИС

В геоинформационном сообществе существует целый ряд ключевых организаций, которые различными методами контролируют деятельность разработчиков ГИС. Наиболее удобным способом такого контроля, на сегодняшний день, становится внедрение и продвижение определенных стандартов, протоколов и RFC разработок. Ниже приводится классификация наиболее активных представителей на рынке web-гис:

- *Ассоциации и регулирующие организации*. OGC (Open Geospatial Consortium) – некоммерческая организация, занимающаяся поддержкой и продвижением стандартов и архитектур, связанных с пространственными данными (например, серии WxS). Членами консорциума являются все более или менее значительные компании, чья деятельность связана с пространственными данными. В стратегическую категорию членов организации входят USGS, NASA, NGA, главными членами являются ESRI, Google, Microsoft и другие.

- *Opensource группы*. OSGeo – также некоммерческая организация, созданная специально для поддержки проектов с открытым исходным кодом. Как правило, поддерживается открытыми сообществами специалистов. Проекты, проходящие инкубацию в OSGeo, получают и место в совете организации. Президентом организации является Frank Warmerdam, создатель и один из основных авторов библиотек GDAL/OGR.

- *Профессиональные ГИС*. ESRI – корпорация, специализирующаяся на ГИС и до недавних пор не имевшая особенных конкурентов. В последнее время ESRI активно пытается усилить свои пошатнувшиеся позиции на рынке web-картографии, развивая ArcGIS Server. Несмотря на наличие на рынке и других игроков (Mapinfo, Autodesk), является стандартом де-факто.

- *Интернет-гиганты*. Google и соратники – группа компаний (включающая также Microsoft, Yahoo и Yandex), рассматривающая web-картографические проекты как один из способов размещения рекламы и активно развивающих online присутствие. В основном популярность достигается за счет предоставления широкому кругу пользователей доступа к ранее недоступным базам данных космической съемки высокого разрешения и сопутствующих технологий маршрутизации и поиска.

- *Генераторы данных*. Поставщики пространственных данных, как правило, коммерческих. Например, поставщики цифровой картографической информации (Navteq/Teleatlas), спутниковых данных (GeoEye, DigitalGlobe). Последнее время в этом секторе появляются и некоммерческие участники (OpenStreetMap).

1.4 Стандарты в области картографических web-сервисов

В настоящее время общие принципы и стандарты в области разработки программного обеспечения, предоставляющего картографические web-сервисы, разрабатываются и декларируются международной некоммерческой организацией *OGC*. OGC была основана 25 сентября 1994 года и на момент создания включала только 8 членов. С 1992 по 2004 год их число возросло с 8 до 250, а на сегодняшний день в OGS представлены наиболее крупные коммерческие, академические и государственные организации, занимающиеся разработкой или исследованиями в области развития и разработки геоинформационного программного обеспечения (в том числе такие крупнейшие корпорации как Boeing, Oracle, ESRI, MapInfo, Intergraph, Google и многие другие).

Во многом деятельность OGC в области геоинформационных систем можно сравнить с деятельностью *W3C* по стандартизации процессов и технологий во всемирной сети. Так, одной из первых разработок OGC были стандарты по созданию GML – *Geography Markup Language* – языка группы XML, предназначенного для описания географически привязанных объектов. GML может быть использован и как язык моделирования, и как язык передачи пространственной информации в сети.

Спецификации OGC предлагают следующие типы картографических web-сервисов [1]:

- *Web Map Service (WMS)*

- определяет параметры запроса и предоставления картографической (пространственной) информации в виде графического изображения или набора объектов;
- описывает условия получения и предоставления информации о содержимом карты (например, свойства объекта в определенном месте на карте);
- характеризует условия получения и предоставления информации о возможностях сервера по представлению различных типов картографической информации.

- *Web Feature Service (WFS)*
  - определяет условия получения и обновления пространственно привязанной информации клиентской частью приложения с использованием Geography Markup Language (GML);
  - описывает стандартный интерфейс доступа и манипуляции с географическими объектами с помощью HTTP-протокола.

- *Web Coverage Service (WCS)*
  - расширяет возможности WMS для предоставления растровой географической информации;
  - в отличие от WMS coverage service разрабатывается для представления свойств и значений в каждой конкретной точке географического пространства, а не на создание готовых картинок, а также позволяет проводить интерпретацию данных не на сервере, а на клиентской части приложения.

1.5 Определение структурных компонентов ИС

Задачу разработки комплексной информационной системы обработки и визуализации геофизических данных можно разделить на две подзадачи: *задачу обработки данных* и *задачу визуализации данных*. Проектировать целостную ИС легче, разделив ее на несколько структурных компонент, тем более что, обработанные данные так или иначе должны где-то храниться и впоследствии визуализироваться в виде цифровой web-карты. То есть, решение, выбранное для хранения обработанных данных, должно иметь возможность интеграции с системой визуализации

данных, такой, как картографический web-сервер. Компоненты, обеспечивающие клиентский доступ к картографическому серверу, будут в полной мере зависеть от типа картографического web-сервиса.

1.6 Задача обработки данных

Обработка данных, для разрабатываемой информационной системы, сводится к трем задачам:

- распаковка сеточных данных (в формате *GRIB или CDF)*, их первичной обработке и сохранения в формате доступном системе визуализации данных;
- обработка данных представленных в табличных видах (координаты, уровни, значения);
- обработка пространственных данных, представленных в виде форматов гео-обмена.

1.6.1 Распаковка сеточных данных

GRIB (Gridded Binary) – математический формат сжатых данных, обычно используемый в метеорологии для хранения исторических и прогнозируемых данных о погоде. Формат был стандартизован комиссией по основным системам *Всемирной Метеорологической Организации* (ВМО), известен под номером GRIB FM 92-IX и описан в 306 номере руководства ВМО по кодам [2]. В настоящее время существуют три версии GRIB. Версия 0 ограниченно используется в таких проектах, как TOGA, и больше практически не используется. Первая редакция (текущая версия — 2) используется во всем мире большинством метеорологических центров для вывода численного прогноза погоды. Новое поколение было введено и известно как вторая редакция GRIB, данные медленно переводятся в этот формат.

Существует несколько программ и библиотек, которые позволяют работать с данными в формате GRIB и конвертировать их. К ним относятся WGRIB, NCAR Command Language, и др. Поскольку задача обработки данных предусматривает перевод данных GRIB в другой формат, с предварительной подготовкой, рассматриваться будут библиотеки, а не утилиты командной строки. Это позволит избежать ошибок, например, перехвата стандартного вывода, и упростить структуру программы обработки данных.

Наиболее популярными библиотеками, обладающими достаточной скоростью и качественной документацией, являются:

- *GRIB_API* - библиотека для раскодирования данных GRIB, разрабатывается и используется центром *European Centre for Medium-Range Weather Forecasts* (ECMWF). Является одной из наиболее полных реализаций стандарта GRIB, позволяет работать с несколькими редакциями форматов, и имеет интерфейсы к языкам программирования C, Python, Fortran 90.

- *G2CLIB* – библиотека для раскодирования данных GRIB, разрабатывается и используется центром *National Centers for Environmental Prediction* (NCEP). Имеет интерфейсы к языкам программирования C, Fortran, Perl и Python. В отличие от GRIB_API, где упор сделан на обработку данных родного центра, имеет более обширную таблицу соответствий кодов, для всех стран участниц ВМО, тем самым, являясь более универсальной библиотекой.

Таким образом, для решения задачи извлечения данных будет использоваться библиотека G2CLIB последней версии.

1.6.2 Сохранение данных

На сегодняшний день, существует несколько основных типов хранилищ пространственной информации:

- *Векторные форматы* – сюда относятся как файловые форматы (ESRI Shapefile, OGR, GML, KML, ArcSDE и др.), так и расширения для работы с пространственными данными СУБД (Oracle Spatial, PostGIS/PostgreSQL, MySQL, MSSQL и др.).

- *Растровые форматы* – JPEG, PNG, TIFF, GeoTIFF и др.

Для хранения динамичной и часто обновляющейся информации в большей степени подходят векторные форматы данных. Их основные преимущества перед растровыми форматами представлены ниже:

- Простота хранения связанной атрибутивной информации об объектах ГИС.

- Масштабирование без потери качества картографического результата, при сравнимых объемах с растровыми форматами.

- Простота создания, особенно при использовании СУБД, типов пространственных данных.
- Возможность контролировать охват, проекции и единицы измерений векторных источников.

Использование растровых форматов целесообразно в следующих случаях:

- При доставке данных аэрокосмических снимков высокой четкости.
- Основы цифровых карт с фиксированными масштабами позволяют снизить нагрузку на картографический сервер.

Часто используется практика сохранения растровых результатов работы картографического сервера и последующее их использование. Такие системы называются кэширующими и выполняют отдачу заранее заготовленных фрагментов цифровой карты. Используются практически во всех крупных проектах (Google Maps, Yandex Maps и др.).

Однозначным выбором для хранения обработанных данных являются векторные форматы хранения. Остается определить, какой тип, файловый или СУБД, удобнее использовать. Если с файловыми векторными форматами вопросов не возникает (стандартом де-факто здесь является формат *ESRI Shapefile*, который, кроме того, указан в постановке задачи данной работы как исходный), то вопрос о хранении пространственных данных в СУБД следует рассмотреть более подробно.

На сегодняшний день, спецификация *OpenGIS* определяет два стандартных способа определения пространственных объектов: в форме *Well-Known Text* (WKT) и в формате *Well-Known Binary* (WKB) [1]. Форматы WKT и WKB включают информацию о типе объекта и координаты, составляющие объект. Примеры текстового представления (WKT) пространственных объектов приведены ниже:

- POINT(0 0)
- LINESTRING(0 0,1 1,1 2)
- POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
- MULTIPOINT(0 0,1 2)
- MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
- MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))

- GEOMETRYCOLLECTION(POINT(2 3),LINESTRING((2 3,3 4)))

Кроме этого, спецификация OpenGIS требует, чтобы внутренний формат хранения пространственных объектов включал идентификатор системы координат (*Spatial Referencing System Identifier* - SRID). Идентификатор SRID необходим для добавления объекта в базу данных. Подавляющее большинство расширений СУБД, предназначенных для хранения в базе географических данных, поддерживают одну из этих, или обе спецификации. Не рассматриваются собственные, не определенные стандартами или спецификациями, форматы.

Теперь можно выделить основные расширения СУБД для хранения пространственных данных:

- *Oracle Spatial* – компонент, обеспечивающий функциональность, необходимую бизнес-приложениям для работы с пространственными данными. Oracle Spatial является опцией к базе данных Oracle 11g Enterprise Edition с полной поддержкой 3D и web-сервисов для управления всеми пространственными данными, в том числе векторными и растровыми данными, топологиями и сетевыми моделями. Является и базируется на коммерческом одноименном продукте, выпускаемом корпорацией Oracle.

- *MySQL* – свободная система управления базами данных, имеет встроенные функции для создания и работы с пространственными данными, но в весьма в ограниченном количестве. Пока не полностью поддерживает спецификации OpenGIS.

- *PostGIS* – расширение объектно-реляционной СУБД PostgreSQL, предназначенное для хранения в базе географических данных. PostGIS включает поддержку пространственных индексов R-Tree/GiST, и функции обработки географических данных. Разрабатывается консалтинговой компанией Refractions Research Inc, как проект в области изучения технологий пространственных баз данных, имеет полную поддержку спецификаций OpenGIS. Является стандартом для практически всех популярных картографических серверов.

- *MSSQL* – коммерческая СУБД, разработанная корпорацией Microsoft. Версии, начиная с Microsoft SQL Server 2008, поддерживают хранение пространственных данных и типов.

- *SQLite Spatial* – расширение, позволяющее работать с пространственными типами данных, для свободной легковесной встраиваемой реляционной базы данных SQLite. SQLite не использует парадигму клиент-сервер, не является отдельно работающим процессом, а предоставляет библиотеку, с которой программа компонуется. Из-за своей файловой

природы, обеспечивает более высокую производительность, при операциях с данными. Но вносит ограничения на количество одновременных подключений, обеспечение и ограничение доступа к БД.

Отдельно необходимо отметить библиотеку *OGR*. Библиотека OGR – это свободная библиотека, предназначенная для работы с векторными данными. Утилиты командной строки, входящие в состав библиотеки, широко используются для выполнения разнообразных задач. Благодаря наличию развитого программного интерфейса, можно работать с функциями OGR из многих языков программирования. Позволяет взаимодействовать с большим количеством векторных форматов, в том числе, и с базами данных. Придерживается спецификаций OpenGIS и имеет собственный SQL-подобный язык. Например, с СУБД SQLite можно взаимодействовать посредством библиотеки OGR, не используя пространственного расширения. Становится возможным хранить пространственные типы данных в простых текстовых форматах, например, CSV (Comma Separated Values – значения, разделенные запятыми). Для взаимодействия с объектами библиотеки OGR, картографический сервер должен иметь ее поддержку.

Из представленной выше информации, можно сделать определенные выводы. Так, обрабатываемую информацию, которая может изменяться со временем, целесообразней хранить в базе данных. Оптимальным же решением, является использование расширения *PostGIS*. Расширение PostGIS обладает подробной документацией, является одним из лидеров в хранении пространственной информации, позволяет масштабировать сервис, и является свободно-распространяемым программным обеспечением.

Выбор решений для задачи хранения пространственных данных выполнен, и будет являться одним из критериев при выборе картографического сервера, для решения задачи визуализации.

1.6.3 Обработка данных

После того, как определены компоненты обеспечивающие извлечение и сохранение данных, можно приступить к выбору инструментов выполняющих их обработку и связующих все вместе. Таким инструментом будет служить язык программирования, который должен удовлетворять следующим требованиям:

- Наличие программного интерфейса к библиотеке G2CLIB.
- Наличие программного интерфейса к СУБД PostgreSQL.

- Наличие библиотек для работы с многомерными массивами и пространственными данными.

- Интерпретируемая природа и возможность манипуляции с высокоуровневыми структурами данных, с акцентом на производительность разработчика.

Программный интерфейс к библиотеке G2CLIB реализован в *Perl* и *Python*. Требованию наличия возможностей математической обработки массивов данных, и библиотек для этого, удовлетворяет в большей степени Python [14]. Его мощное расширение *NumPy*, обладает лучшей производительностью, чем встроенные типы данных Perl. Оба языка имеют прекрасные средства для взаимодействия с PostgreSQL, а наличие поддержки высокоуровневых структур данных, позволит выполнить разработку в короткие сроки.

Наличие в Python такого расширения, как NumPy, делает выбор этого языка программирования наиболее предпочтительным. Определим компоненты, которые будут использоваться для решения задачи обработки данных:

- *Python* – высокоуровневый язык программирования общего назначения с акцентом на производительность разработчика и читаемость кода. Стандартная библиотека языка включает большой объем полезных функций. Язык поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Это позволяет писать на нем как небольшие программы, так и полноценные приложения с использованием объектно-ориентированного подхода.

- *PyNIO* – модуль для языка программирования Python, используемый для чтения и записи файлов в форматах netCDF, GRIB, HDF, HDFEOS2, CCM, и других, широко используемых в геонауках.

- *NumPy* – расширение языка Python, добавляющее поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых математических функций для операций с этими массивами. Поскольку Python интерпретируемый язык, математические алгоритмы часто работают в нем гораздо медленнее, чем в компилируемых языках, таких как C или даже Java. NumPy пытается решить эту проблему для большого количества вычислительных алгоритмов, обеспечивая поддержку многомерных массивов и множество функций и операторов для работы с ними. Таким образом, любой алгоритм, который может быть выражен в основном как последовательность операций над массивами

и матрицами, работает так же быстро как эквивалентный код, написанный на языке программирования C.

- *Psycopg2* – модуль, предоставляющий интерфейс для баз данных PostgreSQL. Соответствует спецификации Python DB API 2.0, поощряющей сходства между модулями Python, используемыми для доступа к базам данных.

Таким образом, задача обработки данных будет решаться с помощью описанных выше инструментов.

1.7 Задача визуализации данных

1.7.1 Выбор типа картографического сервиса

Для выбора решений, обеспечивающих визуализацию пространственных данных, необходимо определить основные критерии, по которым будут отбираться эти решения. Одним из таких критериев, оказывающих очень большое влияние на архитектуру разрабатываемой ИС, является тип картографического сервиса. Типы картографических сервисов, в основном, определены спецификацией OGC, и были рассмотрены выше. Кроме этих типов, существуют решения, позволяющие передавать картографический результат без использования одного из этих стандартов. Например, картографический сервер *MapServer* имеет интерфейс *MapScript*, который обеспечивает доступ к функциям библиотеки MapServer и позволяет создавать картографические приложения на языках высокого уровня (PHP, Python, Perl, .NET и др.). Благодаря этому появляется возможность создавать web-приложения с использованием этих языков программирования, генерировать, обрабатывать и предоставлять пользователю картографический результат. Такого типа расширения (“надстройки”, для MapServer это Chameleon, P.mapper, Fusion, ka-map и др.), над картографическим сервером, имеют несколько весьма значительных недостатков:

- Сложность модификации разработанной информационной системы. При использовании расширения, в будущем невозможно будет сменить один картографический сервер на другой, без смены его клиентской части в виде приложения, использующего это расширение. Таким же образом, довольно трудно будет сменить и клиентскую часть, если

ее возможности, по каким-либо причинам, не устроили конечного пользователя, так как это приведет к перенастройке всей информационной системы.

- Сложность масштабирования. Из-за привязки клиентской части непосредственно к картографическому серверу, невозможно обеспечить горизонтальное масштабирование простыми средствами. Это связано с тем, что каждая копия картографического сервера, предназначенная для расчета картографического результата, должна будет иметь собственный web-интерфейс и сервер к нему, что не только будет снижать производительность и надежность ИС, но и сделает затруднительной работу с атрибутивной информацией картографического сервиса.

- Невозможность использования какой-либо системы, кэширующей картографический результат.

Подобные надстройки (расширения) над картографическими серверами с легкостью оправдывают себя при построении небольших web-сервисов, с небольшим количеством пространственных данных и активных пользователей. Время и простота разработки такой системы значительно ниже, чем системы с использованием одной из спецификаций OGC. Достаточно иметь три структурных компонента: картографический сервер, расширение для него, и HTTP-сервер, обеспечивающий доставку картографического результата пользователю.

В силу вышеперечисленных свойств систем, с использованием и без OGC стандартов, а так же выдвинутых требований к разрабатываемой информационной системе, использование одного из стандартов OGC просто необходимо. В частности, пространственные данные необходимо только визуализировать, без возможности изменения или добавления пользователем атрибутивной информации. На эту роль подходит стандарт *WMS*. Этот стандарт является распространенным, его поддержка реализована во всех популярных картографических серверах. Благодаря использованию этого стандарта, пользователям доступны пространственные данные, множества источников коммерческой и свободной картографической информации (WMS-сервисы Google Maps, Yandex Maps, OpenStreetMap и др.).

1.7.2 Критерии отбора

После выбора типа картографического web-сервиса, можно сформулировать основные критерии, на основании которых будет производиться выбор картографического сервера:

- Открытый исходный код;
- Поддержка стандарта WMS;
- Поддержка векторных типов данных, таких как ESRI Shapefile и PostGIS;
- Поддержка библиотеки PROJ.4 [3], для выполнения манипуляций с картографическими проекциями.

Рассматриваемые продукты будут обязательно соответствовать этим критериям. Дополнительные критерии, которые будут приниматься во внимание, и играют немаловажную роль, но сильно отличаются от продукта к продукту, приводятся ниже:

- Поддержка нескольких растровых форматов для вывода картографического результата. В порядке значимости: PNG, JPEG, GIF. Стоит отметить, что данный список форматов присутствует практически у всех рассматриваемых ниже продуктов, но не каждый из них позволяет производить тонкую настройку формата. Наличие поддержки различных драйверов и библиотек для создания изображения будет плюсом.
- Создание высококачественного картографического результата. Этот критерий напрямую связан с предыдущим, и предусматривает возможность использования сглаживания шрифтов и примитивов географических объектов.
- Производительность картографического сервера. Под производительностью понимается скорость создания изображения цифровой карты. Для оценки данного критерия будут использоваться данные авторитетных источников и существующие исследования.
- Хорошая документированность.
- Простота установки и первоначальной конфигурации.
- Наличие большого сообщества пользователей.

1.7.3 Обзор картографических серверов

Ниже приводится обзор наиболее популярных картографических серверов, обеспечивающих визуализацию пространственных данных, и удовлетворяющих основным критериям, выдвинутым выше.

MapServer

На сегодняшний день MapServer [4] является одной из самых популярных сред создания картографических web-сервисов с открытым исходным кодом. Изначально, MapServer разрабатывался Университетом Миннесоты совместно с Департаментом Природных Ресурсов Штата Миннесота и NASA, а в настоящее время, поддерживается как один из проектов ассоциации OSGeo. Возможность работы MapServer практически на любых платформах (в т.ч. Windows, Linux, Mac OS, Solaris), широчайшие функциональные возможности, легкость интеграции с различными СУБД и открытость исходных кодов, предопределила популярность этого программного продукта. Строго говоря, MapServer позиционируется не как конечное приложение, а как среда разработки (*development environment/platform*). Например, для платформы Windows, MapServer поставляется сконфигурированным серверным комплектом, включающим множество компонентов: Apache HTTP Server, PHP, MapScript, GDAL/OGR, PROJ и др. MapServer является мощным инструментом создания картографических web-сервисов. По своей функциональности не уступает платному программному обеспечению, а по части легкости настройки и интеграции с СУБД, превосходит многие из них. К основным достоинствам программы можно отнести следующие:

- Возможность работы на большинстве популярных платформ;
- Поддержка большого количества растровых и векторных форматов;
- Полное соответствие стандартам, разработанным OpenGIS Consortium, в отношении web mapping services (поддержка WMS, WFS, WCS);
- Возможность переконфигурирования и программирования с использованием Perl, PHP, Java, C, Python и др.;
- Возможность интеграции с Oracle, Sybase, MySQL, PostgreSQL и другими СУБД;
- Создание высококачественного картографического результата (поддержка TrueType шрифтов, масштабируемых подписей, раскрасок, экспорт в PNG, TIFF, GIF, JPEG форматы);
- Полностью открытый бесплатный компилируемый код на C и мощная пользовательская поддержка.

Расплатой за функциональность является тот факт, что MapServer не является простой средой разработки. Для создания полнофункционального сайта, потребуются знания одного или

нескольких перечисленных выше языков программирования и хорошие навыки администрирования. Тем не менее, данный программный продукт, полностью удовлетворяет выдвинутым критериям и прекрасно подходит для разработки системы визуализации данных.

GeoServer

Так же, как и MapServer, GeoServer [5] является картографическим сервером с открытым исходным кодом, который, среди многих прочих возможностей, реализует следующие спецификации OGC: WMS, WFS, WCS. Однако, в отличие от MapServer, GeoServer реализует спецификацию WFS-T (WFS-Transaction). Это означает, что, используя GeoServer, возможно не только получать данные для построения на их основе собственных карт, но также редактировать полученные данные с последующим автоматическим обновлением исходной информации на сервере. Среди поддерживаемых форматов значатся: JPEG, PNG, SVG, KML/KMZ, GML, PDF, ESRI Shapefile и др..

Другой интересной особенностью, отличающей GeoServer от MapServer, является, поставляемая с GeoServer, визуальная система управления файлами настроек и описания данных. Эта система реализована в виде web-интерфейса и предоставляет пользователю возможность интерактивного создания и изменения, разрабатываемого картографического сервиса. GeoServer создан на основе Geotools – набора инструментов, разработаных на Java. Для его работы необходимо иметь установленный в системе *Java Development Kit* (JDK). Следует отметить, что GeoServer является основным и серьезным конкурентом MapServer с похожим пакетом функциональности. Основным недостатком GeoServer является несколько худшее быстродействие и требовательность к ресурсам сервера в следствии использования языка программирования Java (в противовес Си у MapServer)

MapGuide Open Source

MapGuide Open Source – является web-платформой, которая позволяет пользователям разрабатывать и развертывать web-картографические сервисы и приложения. Это открытая версия приложения Autodesk MapGuide Enterprise. Характерные особенности MapGuide: интерактивный просмотр результата, который включает в себя поддержку выбора характеристик, свойств, легенды карты, а так же инструментов типа “буфер”, линеек и систем измерения; включает в себя базу данных XML, для управления всем содержимым и настройками, а также поддерживает большинство популярных пространственных форматов данных. Имеет возможности для развертывания на Windows и Linux, поддерживает работу с Apache и IIS HTTP-серверами, и предлагает поддержку широкого спектра языков программирования (PHP, .NET, Java, JavaScript) и

программных интерфейсов, для разработки приложения. Картографический сервер MapGuide Open Source распространяется под лицензией LGPL.

MapGuide OpenSource позволяет с легкостью разворачивать web-сервис и получать при этом прекрасный картографический результат. Этот программный продукт включает в себя прекрасный графический интерфейс управления настройками и содержимым, позволяющий создавать из готовых элементов web-интерфейс пользователя. По сути, является законченным продуктом, и позволяет в очень короткие сроки разворачивать картографический сервис. К сожалению, при работе с несколькими слоями пространственных данных проект показывает не самую высокую производительность. Взаимодействие с источниками данных производится с помощью библиотеки FDO (*Feature Data Object*). Каких-либо стабильных результатов работы с ней удалось добиться лишь при использовании закрытого формата векторных данных SDF. Сказывается проприетарная родословная проекта. Стоит отметить и тяжелый процесс установки из исходных кодов, который требует хорошего уровня подготовки разработчика (вплоть до редактирования исходных текстов программных компонентов), и не всегда его успешность. Для решения поставленной задачи визуализации данных, данный программный продукт не пригоден.

Mapnik

Mapnik – открытая библиотека для создания растровых карт из векторных данных. Библиотека разработана для проекта *OpenStreetMap,* и используется другими достаточно крупными ресурсами. Основная заявленная цель Mapnik – получение красивых карт. Для использования, необходимо написать программу на Python, вызывающую нужные функции библиотеки. Mapnik конкурирует с MapServer в части создания качественных растровых карт из векторных данных. Оба продукта используют при отрисовке линий, полигонов и шрифтов драйвер *AGG* и *FreeType* с *anti-aliasing*. В целом, Mapnik явно разрабатывался под новомодную стилистику Google Maps, и пока проигрывает по настраиваемости и качеству документации MapServer. Также следует отметить необходимость фактически “писать” свой картографический сервер, что сильно повлияет на время разработки. Данные ограничения делают этот программный продукт непригодным для решения поставленной задачи.

1.8 Выводы

В данном разделе была проведена частичная декомпозиция исходной задачи на подзадачи. Каждая из подзадач была детально рассмотрена, и для каждой были выбраны основные

программные компоненты, из которых будет состоять разрабатываемая информационная система. Для определения картографического сервера, пригодного для решения задачи визуализации данных, требуется провести сравнительное исследование двух программных продуктов: MapServer и GeoServer в свете решения задачи визуализации гео-привязанных данных в метеорологии.

2 Исследование и построение решения задачи

В данном разделе определяются требования к функциональности, и проводится сравнительное исследование двух картографических серверов: MapServer и GeoServer. По результатам проведенного исследования определяется архитектура разрабатываемого сервиса и выполняется построение решения задачи.

2.1 Определение требований

Для проведения корректного сравнения и исследования картографических серверов, и выбора наиболее подходящего из них, необходимо определить условия и критерии, по которым эти картографические сервера будут сравниваться. Основные критерии, по которым будут сравниваться программные продукты, следующие:

- Производительность;
- Простота настройки.

Под производительностью картографического сервера, понимается скорость создания и отдачи картографического результата, или скорость создания изображения из исходных пространственных данных для заданных параметров. Такими параметрами могут быть, например, использование определенной проекции, определенного охвата цифровой карты, формат результирующего изображения и др. Далее приводятся все необходимые параметры и данные, неосвященные в предыдущем разделе, с оглядкой на стандарт WMS.

2.1.1 Используемые возможности WMS

WMS - является протоколом осуществления запроса к картографическому серверу, и получения результата от него. Так же, протокол позволяет посредством запросов *GetFeatureInfo* получать информацию об объектах ГИС. В ходе исследования и определения производительности, запросы такого типа использоваться не будут.

Рассматривать запросы типа GetFeatureInfo не имеет смысла потому, что они покажут скорость получения атрибутивной информации из хранилища пространственных данных. Такой запрос не требует значительных вычислительных ресурсов, так как содержит в ответе текстовую информацию, и опирается, в основном, на производительность хранилища и интерфейса связи с ним. Например, векторный формат Shapefile, содержит атрибутивную информацию об объектах ГИС в формате хранения данных DBF, время выполнения запроса к которому, будет существенно меньше скорости создания изображения участка карты. К тому же, ответ на запрос GetFeatureInfo может быть представлен в одном из нескольких форматов, рассмотрение которых имеет смысл для конкретного картографического сервера на этапе разработки клиентского приложения.

2.1.2 Подготовка исходных данных

В качестве исходных данных для выполнения данной работы были предоставлены пространственные данные в формате Shapefile. Эти данные содержат в себе слои с информацией об объектах ГИС. Слои включают в себя разное количество и объем пространственной информации и, соответственно, различаются размером. Ниже представлено описание нескольких слоев, необходимых для проведения исследования:

- Основа (“подложка”) цифровой карты. Для создания нагрузки будет использоваться самый объемный слой (около 160Мб) hdrlin, содержащий данные о гидрологических объектах; для быстрой оценки состояния карты будет использоваться “легкий” (около 500Кб) слой crdlin, содержащий координатную сетку.

- В качестве динамичных данных, выбран слой poppnt. Данный слой выбран потому, что имеет тип пространственных данных POINT, который используется и для представления сетки GRIB обрабатываемых данных.

Предварительно, необходимо подготовить слой poppnt. Сделать это можно воспользовавшись утилитой `shp2pgsql`, входящей в состав библиотеки GDAL. Ниже представлен скрипт `convert.sh`, позволяющий конвертировать данные слоя в SQL-дамп для PostGIS:

```
for f in *shp
do
  # удалить расширение и заменить дефисы на подчеркивания
  name=$(basename $f .shp| tr "-" "_")
```

```
        # конвертировать файл $f из shp в набор команд sql
        shp2pgsql –d –I –s 4326 $f $name > ./sql/$name”.sql”
done
```

2.1.3 Картографические проекции

Картографическая проекция – математически определенный способ отображения поверхности эллипсоида на плоскости. Суть проекций связана с тем, что фигуру Земли – эллипсоид, не развертываемый в плоскость, заменяют на другую фигуру, развертываемую на плоскость. При этом с эллипсоида на другую фигуру переносят сетку параллелей и меридианов. Вид этой сетки бывает разный в зависимости от того, какой фигурой заменяется эллипсоид.

Библиотекой, позволяющей картографическим серверам выполнять расчеты проекций, является *PROJ.4*. Эта библиотека поддерживает специальную нотацию, с помощью которой можно задать правила необходимых преобразований, нотация достаточно сложна и объемна, а так же требует определенных знаний в области геодезии и картографии, поэтому в данной работе не приводится.

Представленные выше исходные данные хранятся в так называемой, *неспроецированной* (географической, широта/долгота) системе координат, имеющей код *EPSG:4326*. Для перевода их в спроецированную систему, которую описывает код *EPSG:3576*, следует определить несколько нотаций, которые смогут использовать картографические сервера и библиотека PROJ.4 для манипуляций с данными:

- *Для библиотеки PROJ.4:*

```
+proj=laea +lat_0=90 +lon_0=90 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84
+units=m +no_defs
```

- *Для MapServer:*

```
PROJECTION
"proj=laea"
"lat_0=90"
"lon_0=90"
"x_0=0"
"y_0=0"
```

```
"ellps=WGS84"

"datum=WGS84"

"units=m"

"no_defs"

END
```

- *Для PostGIS:*

```
INSERT into spatial_ref_sys (srid, auth_name, auth_srid, proj4text,
srtext) values ( 93576, 'epsg', 3576, '+proj=laea +lat_0=90 +lon_0=90
+x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m +no_defs ',
'PROJCS["WGS 84 / North Pole LAEA Russia",GEOGCS["WGS
84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6
326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0
1745329251994328,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]],UN
IT["metre",1,AUTHORITY["EPSG","9001"]],PROJECTION["Lambert_Azimuthal_Eq
ual_Area"],PARAMETER["latitude_of_center",90],PARAMETER["longitude_of_c
enter",90],PARAMETER["false_easting",0],PARAMETER["false_northing",0],A
UTHORITY["EPSG","3576"],AXIS["X",UNKNOWN],AXIS["Y",UNKNOWN]]');
```

- *Для GeoServer:*

```
3576=PROJCS["WGS 84 / North Pole LAEA
Russia",GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",63
78137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.01745329251
9943295]],PROJECTION["Lambert_Azimuthal_Equal_Area"],PARAMETER["latitud
e_of_center",90],PARAMETER["longitude_of_center",90],PARAMETER["false_e
asting",0],PARAMETER["false_northing",0],UNIT["Meter",1]]
```

Представленные параметры добавляют поддержку проекции с кодом EPSG:3576 тому или иному программному продукту. Обычно, для добавления поддержки этого кода, необходимо добавить строку с параметрами проекции в базу данных кодов EPSG, которой, в большинстве случаев, является одноименный файл.

Картографический результат должен предоставляться в нескольких проекциях. Перепроецирование данных на этапе подготовки, сделает очень тяжелым поддержку хранилищ пространственных данных и их добавление впоследствии, так как придется создавать несколько конфигураций для картографического сервера. Также этот недостаток сделает трудоемким процесс горизонтального масштабирования информационной системы. Что бы избежать этого недостатка, следует настроить картографический сервер на использование нескольких проекций. После чего, картографический результат в нужной проекции можно будет получить, указав специальный параметр WMS. Но у этого подхода имеется один недостаток: повышенное использование системных ресурсов при перепроецировании данных “на лету” картографическим сервером, что приводит к некоторому снижению производительности.

Тем не менее, использоваться будет именно второй подход, так как он позволяет настроить хранилище пространственных данных один раз, и избежать необходимости его перенастройки при использовании отличной от исходной проекции.

2.1.4 Охват карты

Охват определяет ограничивающий прямоугольник карты. Значения параметров охвата вводятся в соответствии с форматом: нижняя левая координата X; нижняя левая координата Y; верхняя правая координата X; верхняя правая координата Y. Эти значения должны быть в тех же единицах измерения, что и пространственные данные. Для каждой используемой проекциитребуется определить свой охват. Так как в распоряжении имеется слой, содержащий географическую сетку (crdlin), то охват карты для каждой проекции будет определяться по нему.

Для определения охвата файла `crdlin.shp`, можно воспользоваться утилитой `ogrinfo`, которая входит в библиотеку GDAL/OGR. Определим охват данных в географической системе координат EPSG:4326:

```
ogrinfo –al –so crdlin.shp
```

Вывод команды:

```
 INFO: Open of `crdlin.shp'
      using driver `ESRI Shapefile' successful.

Layer name: crdlin
Geometry: Line String
Feature Count: 701
Extent: (-180.000000, 40.000000) - (180.000000, 84.000000)
Layer SRS WKT:
GEOGCS["GCS_WGS_1984",
    DATUM["WGS_1984",
        SPHEROID["WGS_1984",6378137,298.257223563]],
    PRIMEM["Greenwich",0],
    UNIT["Degree",0.0174532925199433295]]
SHAPE_Leng: Real (19.11)
class_id: Integer (9.0)
```

```
lon: Real (19.11)
lat: Real (19.11)
```

Для того, что бы получить охват пространственных данных слоя crdlin в проекции EPSG:3576, предварительно, этот слой необходимо перепроецировать. Для этого, можно воспользоваться утилитой `ogr2ogr` библиотеки GDAL/OGR:

```
ogr2ogr -t_srs "+proj=laea +lat_0=90 +lon_0=90 +x_0=0 +y_0=0 +ellps=WGS84
+datum=WGS84 +units=m +no_defs" crdlin2.shp crdlin.shp
```

После, повторить команду, которая даст нужную информацию об охвате:

```
ogrinfo -al -so crdlin2.shp
```

Вывод команды:

```
INFO: Open of `crdlin2.shp'
      using driver `ESRI Shapefile' successful.

Layer name: crdlin2
Geometry: Line String
Feature Count: 701
Extent: (-5133549.567108, -5397733.446068) - (5397733.446068, 1667991.365941)
Layer SRS WKT:
PROJCS["Lambert_Azimuthal_Equal_Area",
    GEOGCS["GCS_WGS_1984",
        DATUM["WGS_1984",
            SPHEROID["WGS_1984",6378137,298.257223563]],
        PRIMEM["Greenwich",0],
        UNIT["Degree",0.0174532925199433295]],
    PROJECTION["Lambert_Azimuthal_Equal_Area"],
    PARAMETER["latitude_of_center",90],
    PARAMETER["longitude_of_center",90],
    PARAMETER["false_easting",0],
    PARAMETER["false_northing",0],
    UNIT["Meter",1]]
SHAPE_Leng: Real (19.11)
class_id: Integer (9.0)
```

```
lon: Real (19.11)
lat: Real (19.11)
```

Таким образом, мы получили искомые значения охвата для обеих проекций:

- *EPSG:4326* - Extent: (-180.000000, 40.000000) - (180.000000, 84.000000)
- *EPSG:3576* - Extent: (-5133549.567108, -5397733.446068) - (5397733.446068, 1667991.365941)

2.1.5 Формат вывода

Выбранные для исследования картографические сервера поддерживают следующие растровые форматы вывода:

- *PNG* (Portable Network Graphics) – растровый формат хранения графической информации, использующий сжатие без потерь по алгоритму *Deflate*
- *JPEG* (Joint Photographic Experts Group) – один из популярных графических форматов, применяемый для хранения фотоизображений и подобным им изображений, алгоритм JPEG позволяет сжимать изображение как с потерями, так и без потерь
- *GIF* (Grpahics Interchange Format) – популярный формат графических изображений, способен хранить сжатые данные без потери качества в формате не более 256 цветов, использует формат сжатия LZW

Формат PNG является самым новым форматом, имеющим большую популярность для web. Формат появился как замена GIF и, частично, TIFF. Использование этого формата дает преимущество в виде наилучшего соотношения объем/качество изображения, что делает его оптимальным выбором для представления цифровых карт. Так же, PNG обладает рядом других характеристик, которые делают его привлекательным для использования в качестве основного формата вывода картографического изображения:

- Полноценная поддержка alpha-transparency – прозрачности. Позволяет сделать участки изображения прозрачными и полупрозрачными.
- Качественный алгоритм сжатия без потери качества, похожий на алгоритм LZW, но намного более эффективный.

- Возможность чересстрочной развертки, причем (в отличие от GIF) как по вертикали, так и по горизонтали одновременно.

- Встроенная гамма-коррекция. Позволяет прикрепить к изображению настройки его отображения, для того чтобы на разных мониторах изображение отображалось точно так же, как и при создании.

Часто картографические сервера позволяют выбирать драйвер (библиотеку), с помощью которого осуществляется создание изображения. Чаще всего, используется библиотека GD (*Graphics Library*) – программная библиотека, для динамической работы с изображениями. Библиотека позволяет создавать изображения в форматах GIF, JPEG, PNG и WBMP, состоящие из линий, дуг, текста (включая программный выбор шрифтов) и других изображений, а также использовать различные цвета. В последних версиях библиотеки добавлена поддержка 32-битных (*truecolor*) изображений и многое другое. На рисунке 2.1 представлен пример изображения участка карты, созданного с помощью библиотеки GD.

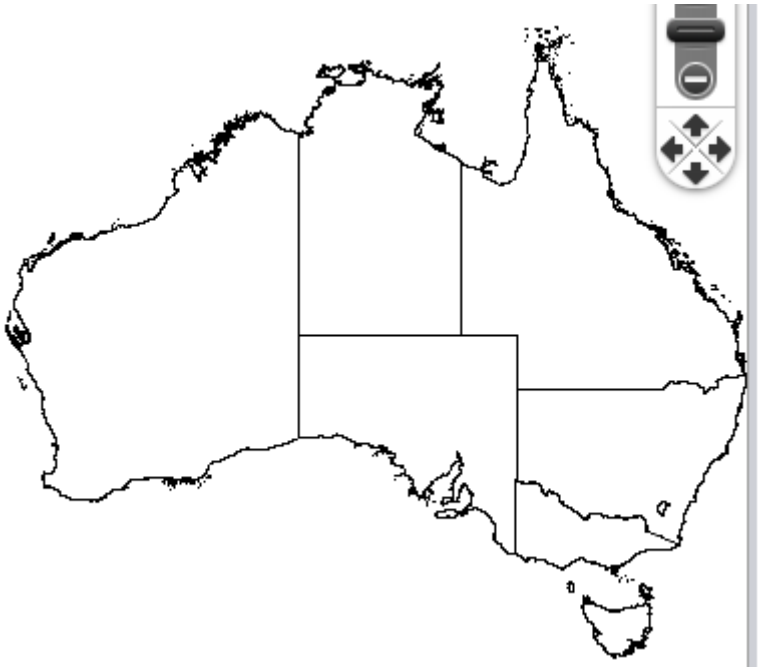


Рисунок 2.1 – Изображение участка карты, созданного с помощью GD

Из представленного выше примера видно, что очертания карты достаточно “резки”. При использовании GD, трудно достичь желаемого уровня сглаживания. Способы, которые предоставляет эта библиотека, не всегда применимы для сферы картографии (например, уменьшение изображения в четное количество раз с использованием фильтрации).

Относительно недавно картографические сервера обзавелись поддержкой другой библиотеки – AGG (*Anti-Grain Geometry*). Эта библиотека предоставляет графический

инструментарий для создания высококачественных изображений 2D. Библиотека AGG изначально была создана для проектов, которые требуют высококачественного графического результата, например, географических карт. При этом AGG не является обычной графической библиотекой, а позиционируется как “инструмент для создания других инструментов”, поэтому достаточно сложна в использовании. На рисунке 2.2 представлено сравнение качества изображений созданных с помощью GDI+ (*Graphics Device Interface*) и AGG.

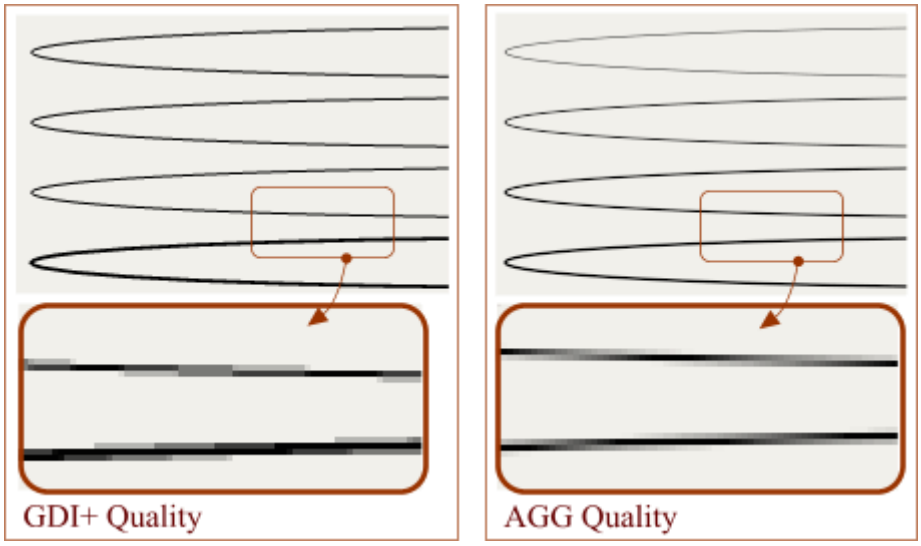


Рисунок 2.2 – Сравнение качества изображений созданных с помощью GDI и AGG

Для сравнения с библиотекой GD, на рисунке 2.3 приводится изображение участка карты созданного с помощью инструментов AGG.

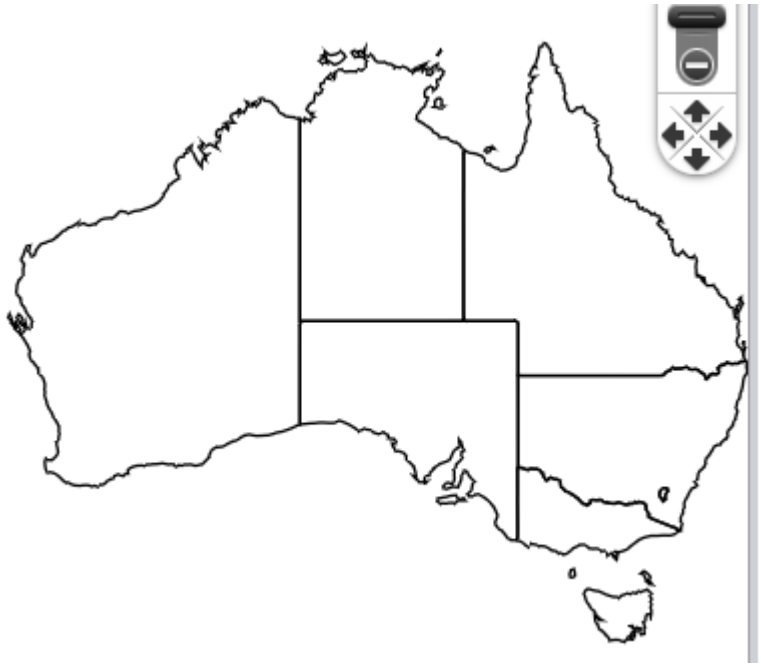


Рисунок 2.3 – Изображение участка карты, созданного с помощью AGG

Представленный результат, по сравнению с результатом, созданным GD, имеет прекрасное качество. Линии сглажены и плавны, и именно такой результат картографического изображения отвечает ранее поставленным требованиям.

Из представленных выше примеров отлично видно, что библиотека AGG, вместе с использованием формата PNG, обеспечивает высокое качество картографического результата. Именно эти решения будут использоваться при проведении исследования и разработке системы визуализации данных.

2.2 Тестовая платформа

При проведении исследования и разработке необходимо будет настраивать и сравнивать, множество различных программных продуктов между собой. При этом возникает ряд проблем, которые могут негативно повлиять на полученные в ходе экспериментов результаты, и на общую производительности системы в целом. Самой главной проблемой является неоднородность среды, в которой проводится исследование. Эта неоднородность может проявляться в различных установленных системных параметрах, настройках рассматриваемых программных продуктов, в системных сервисах, которые они используют. Что бы обеспечить однородность среды, в которой будет проводиться исследование, необходимо эту среду сохранить в каком-то первоначальном состоянии – “снимке”, и при первой необходимости, иметь возможность вернуться к нему.

Существует несколько способов создания “снимков”, или образов, системы. При этом не только сохраняются все файлы и настройки операционной системы, вместе с пользовательскими данными, но и сохраняется структура разделов жесткого диска, если такой образ создавать для всего устройства. Существует множество программ для создания снимков состояния операционной системы. Для платформы Windows, например, существует платное программное обеспечение Acronis True Image. Для платформы Linux, при использовании технологии LVM (*Logical Volume Manager,* - менеджер логических томов), также существует возможность создавать снимки целых разделов. Данный подход обеспечивает сохранение однородности тестовой платформы, но имеет несколько существенных недостатков, которые для проведения исследования являются критичными:

- Необходимость наличия нескольких компьютеров, одного - для запуска тестовой платформы, другого - для снятия результатов.

- Необходимость предварительной подготовки системы к созданию и восстановлению образа.
- Необходимость наличия хранилища большого объема для создаваемых образов, особенно, если используемое программное обеспечение не поддерживает функцию создания дифференциальных или инкрементных архивов.
- Зависимость от используемой аппаратной части.

Каждый из этих недостатков также может иметь дополнительные негативные последствия для результатов исследования. Например, неоднородность сетевой среды между тестовым компьютером и компьютером для снятия результатов. Чтобы избежать таких негативных последствий и недостатков, применяются *технологии виртуализации*.

2.2.1 Технология виртуализации

Виртуализация в вычислениях – процесс представления набора вычислительных ресурсов, или их логического объединения, который дает какие-либо преимущества перед оригинальной конфигурацией. Виртуализация – это общий термин, охватывающий абстракцию ресурсов для многих аспектов вычислений.

В рассматриваемой задаче, использование технологии виртуализации дает следующие преимущества:

- Легкость разработки и тестирования. Множество продуктов виртуализации позволяют запускать несколько различных операционных систем одновременно, позволяя тем самым разработчику программного обеспечения тестировать его приложение на нескольких платформах. Также, удобные средства по созданию “снимков” текущего состояния системы одним кликом мыши, и такого же простого восстановления из этого состояния, позволяют создавать тестовые окружения для различных конфигураций, что существенно повышает скорость и качество разработки.
- Возможность консолидации серверов. Приложения, работающие на серверах в IT-инфраструктуре компаний, создают небольшую нагрузку на аппаратные ресурсы серверов. Виртуализация позволяет загрузить, или консолидировать, такие физические сервера

виртуальными, тем самым повысив коэффициент использования аппаратуры, что позволяет существенно сэкономить и на аппаратуре, и на обслуживании, и на электроэнергии.

На данном этапе наиболее интересно первое преимущество, в то время как второе, будет рассматриваться в рекомендациях по масштабированию системы.

Существует несколько типов виртуализации, рассмотрим подробнее каждый из них:

- *Программная виртуализация*

    a. *Динамическая трансляция*. При динамической трансляции проблемные команды гостевой ОС перехватываются гипервизором (программа или аппаратная схема, обеспечивающая или позволяющая одновременное, параллельное выполнение нескольких или даже многих операционных систем на одном и том же *хост-компьютере*). После того как эти команды заменяются на безопасные, происходит возврат управления гостевой ОС.
    b. *Паравиртуализация*. Техника виртуализации, при которой гостевые ОС подготавливаются для исполнения в виртуализированной среде, для чего их ядро незначительно модифицируется. Данный метод позволяет добиться более высокой производительности, чем при динамической трансляции.

2. *Аппаратная виртуализация*. Позволяет запускать на одном физическом компьютере (*хосте*) несколько операционных систем (*гостевых* ОС) в целях обеспечения их независимости друг от друга.

После разработки информационной системы, тестовая платформа должна иметь возможность простой установки на аппаратные и виртуализированные платформы. Такое требование может удовлетворить аппаратная виртуализация. При ее использовании нет ограничений на совместимость платформы *хостовой* и *гостевой* ОС, а так же не требуется модификация ядра ОС.

Аппаратная виртуализация относится к виртуализации платформ, продуктом которой, являются *виртуальные машины* – некие программные абстракции, запускаемые на платформе реальных аппаратно-программных средств. Аппаратная виртуализация подразумевает использование одной из технологий эмуляции используемого CPU (Central Processing Unit, - центральное обрабатывающее устройство), например для CPU производства Intel такая технология носит название *VT*, а для CPU производства AMD название *AMD-V*.

Тестирование и исследование быстродействия проводилось на базе хост компьютера с указанными ниже характеристиками[1]:

- Тип ЦП: Mobile DualCore Intel Core 2 Duo P8400, 2266 MHz (8.5 x 267) поддерживающий технологию Intel Virtualization Technology (VT-x)
- Доступная системная память: 3066 Мб (DDR2-800 DDR2 SDRAM)
- Чипсет системной платы: Intel Cantiga PM45
- Операционная система: Microsoft Windows 7 Ultimate 32 bit (MSDN License)
- Дисковый накопитель: Seagate FreeAgent GoFlex USB Device (465 Гб, USB)

Отдельно, стоит сказать про указанный дисковый накопитель. Для уменьшения влияния хостовой ОС, виртуальная машина с гостевой ОС будет размещена на отдельном носителе. Этот носитель, хоть и обладает более низкой производительностью по сравнению с настольными решениями, позволит провести исследование более корректно.

Чтобы иметь примерное представление о скоростных характеристиках внешнего дискового накопителя, воспользуемся бесплатным тестовым приложением *CrystalDiskMark*, предназначенным для сравнительного анализа и тестирования жестких дисков компьютера на скорость чтения и записи в разных режимах. Результаты этого теста приведены на рисунке 2.4.

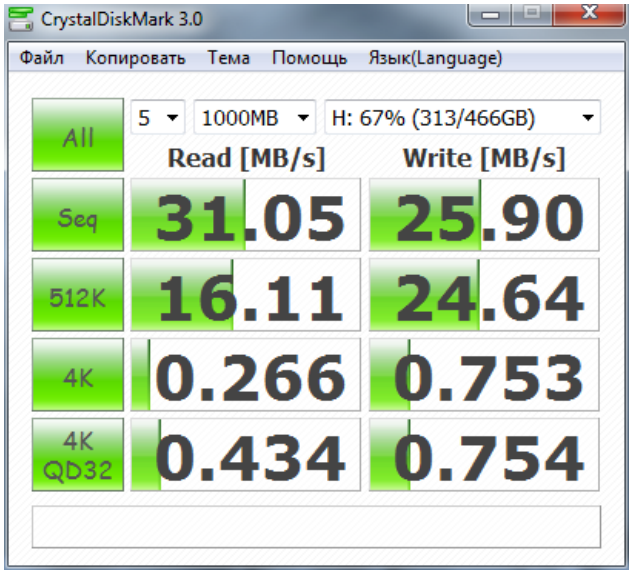


Рисунок 2.4 – Скоростные характеристики внешнего накопителя

Основные характеристики и условия использования хост-компьютера определены, и можно приступить к выбору программного обеспечения для виртуализации. На сегодня, на рынке представлено два лидера, выпускающих решения для рабочих станций, и имеющих версии для ОС Windows 7:

[1] Внедрение на сервере sibnigmi.ru осуществлено на базе хост-системы Rstyle Marshall Xeon под управлением операционной системы CENTOS и системой виртуализации KVM.

- *VMware Workstation* – позволяет создавать и запускать одновременно несколько виртуальных машин (x86-архитекутры), в каждой из которых работает своя гостевая операционная система. Поддерживает как 32-битные, так и 64-битные ОС. Является платным программным обеспечением с ознакомительным периодом 30 дней.

- *VirtualBox* – программный продукт, обеспечивающий виртуализацию, для операционных систем Microsoft Windows, Linux, FreeBSD, Mac OS X и других. Базовая версия полностью открыта по лицензии GNU GPL, соответственно, не имеет ограничений на использование.

Несмотря на то, что VirtualBox является свободно-распространяемым программным обеспечением, для создания тестовой платформы будет использован программный продукт VMware Workstation. Есть несколько причин, обуславливающих такое решение, которое полностью является личным мнением автора данной работы, сформированным в процессе использования обоих программных продуктов:

- Более стабильная работа VMware Workstation в фоновых режимах, в т.ч. режимах сна.

- Расширенные, по сравнению с VirtualBox, сетевые возможности VMware Workstation.

- Ознакомительный период VMware Workstation, дает право создавать виртуальную машину без ограничений. Далее, можно воспользоваться программой VMware Player – бесплатным (для личного некоммерческого использования) программным продуктом, предназначенным для создания и запуска готовых виртуальных машин (в т.ч. и созданных в VMware Workstation).

Процесс установки программы VMware Workstation тривиален, и в данной работе рассматриваться не будет. На рисунке 2.5 представлено окно VMware Workstation после первого запуска.

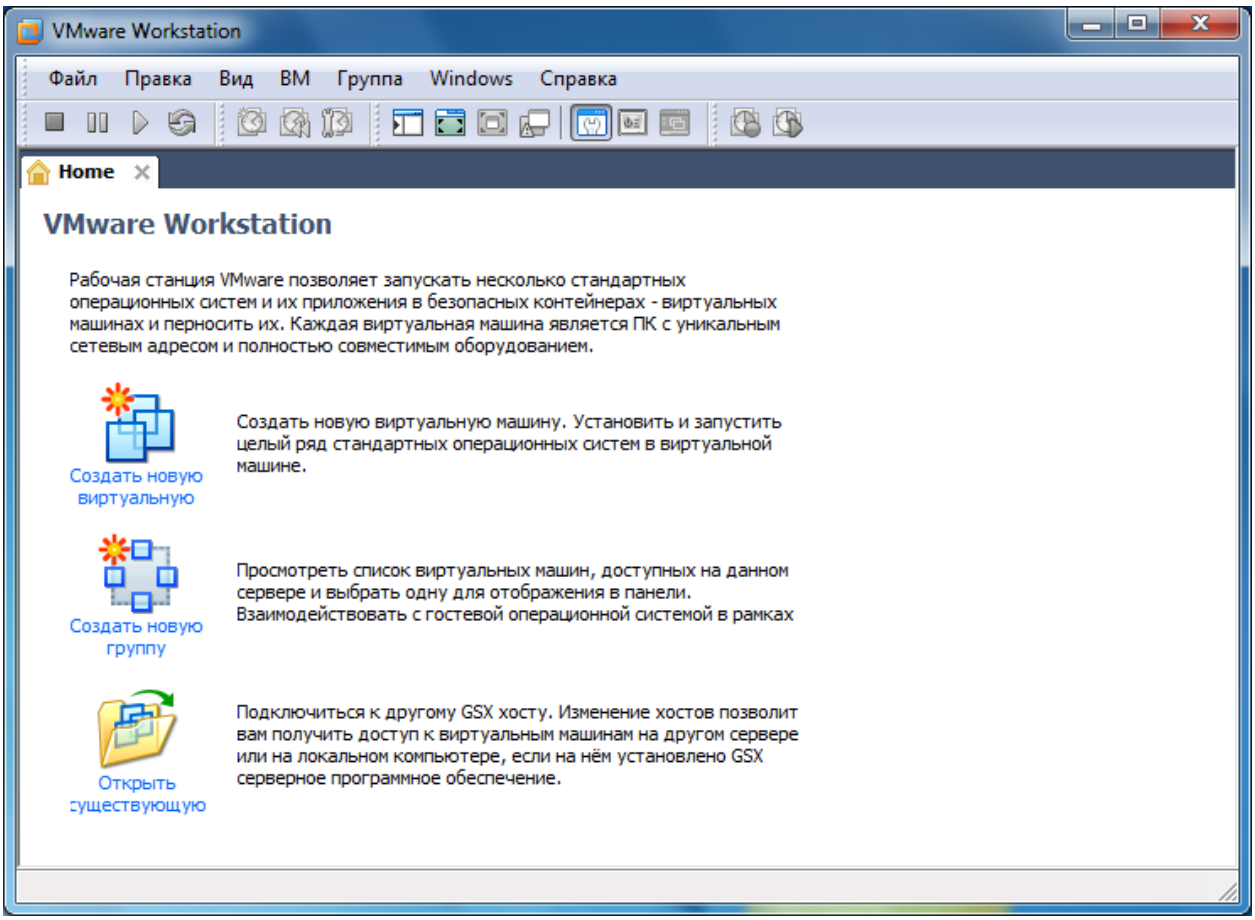


Рисунок 2.5 – Главное окно VMware Workstation

Программа имеет интуитивно-понятный интерфейс, и каких-либо сложностей, в процессе работы, вызывать не должна. Вызвав одноименный пункт меню, можно приступить к созданию виртуальной машины. Создание виртуальной машины заключается в ответах на вопросы мастера, предоставляющего необходимую информацию о тех или иных параметрах. Ниже описаны шаги создания виртуальной машины для тестовой платформы:

- Configuration: Custom (advanced)
- Hardware compatibility: Workstation 6.5-7.0
- I will install the operating system later
- Linux: Red Hat Linux
- Virtual machine name: wmsystem
- Number of processors: 1; Number of cores per processor: 1
- Memory: 2048 MB

- Use network address translation (NAT) – использование этого режима обеспечит доступ виртуальной машины в интернет, для обновления ОС и установки необходимых пакетов
- SCSI Adapter: BusLogic
- Disk: Create a new virtual disk
- Virtual disk type: SCSI
- Maximum disk size (GB): 37; Store virtual disk as a single file; Allocate all disk space now – последняя опция нужна для того, чтобы распределить дисковое пространство для файла жесткого диска сразу, что положительно сказывается на фрагментации
- Disk file: wmsystem.vmdk
- Finish

На рисунке 2.6 представлено окно созданной виртуальной машины, с описанной выше конфигурацией.

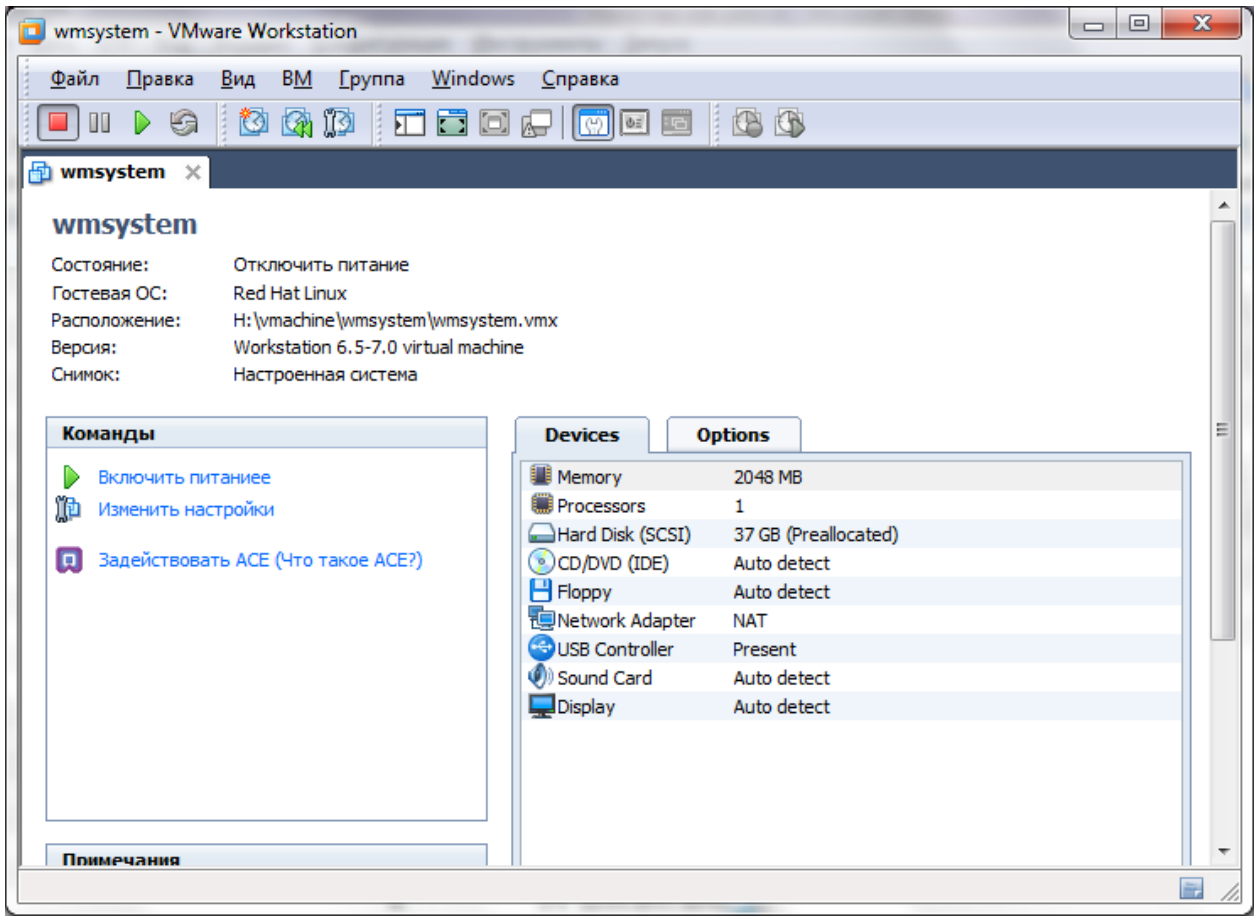


Рисунок 2.6 – Окно созданной виртуальной машины

2.2.2 Выбор и настройка дистрибутива ОС

Для завершения подготовки тестовой платформы, необходимо выбрать и установить дистрибутив ОС. Критерии, по которым будет производиться выбор дистрибутива операционной системы, представлены ниже:

- Дистрибутив ОС со свободной лицензией.
- Дистрибутив, основанный на пакетах программного обеспечения.
- Наличие большого сообщества пользователей.
- Наличие дополнительных хранилищ пакетов программного обеспечения (репозиториев), необходимых для поиска и установки всех необходимых программных продуктов из области ГИС.
- Наличие последних версий программного обеспечения в репозиториях.

Таким критериям, на сегодняшний день, максимально соответствуют два дистрибутива ОС *Linux*: *Fedora* и *Ubuntu*. Оба дистрибутива ОС чрезвычайно популярны у широкого круга пользователей, имеют массу пакетов свежего программного обеспечения, и конкурируют между собой. Из-за патентных ограничений США в дистрибутив Fedora не включаются такие программы, как кодеки мультимедиа, драйверы видеокарт, некоторые научные пакеты, и др. Этот факт имеет положительное влияние, потому что в дистрибутив включаются программы только с полностью свободной лицензией, не имеющей никаких оговорок в части использования. Пользователь сам решает, что и с какими ограничениями он может добавить после установки дистрибутива. Автор данной работы имеет большой опыт настройки и использования дистрибутивов, основанных на RPM (*Red Hat Package Manager*), коим является и дистрибутив Fedora, поэтому предпочтение отдается ему.

Проект Fedora задуман компанией Red Hat, как тестовая площадка для новых технологий и компонентов систем, которые позднее могут быть использованы в корпоративных дистрибутивах. Это является одновременно и достоинством и недостатком данного дистрибутива. Дело в том, что дистрибутив очень стремительно развивается, новые версии выходят раз в полгода, следовательно, он содержит самые последние версии пакетов программного обеспечения. Что, с оглядкой на тематику и направление задачи, является достоинством. Научные пакеты в корпоративных дистрибутивах, таких как *CentOS* (полный совместимый аналог Red Hat Linux), обновляются

крайне редко, а собирать программное обеспечение из исходных текстов для пакетных дистрибутивов, является плохой практикой.

Процесс установки данного дистрибутива тривиален, и в данной работе рассматриваться не будет. Несколько отличий, от предлагаемых вариантов установки, заключаются в выборе минимальной комплектации дистрибутива пакетами, и упрощенного разбиения диска: для раздела подкачки (*swap*) отведено 2047 MB, остальное пространство отведено под корневой (*/*) раздел. Такое разбиение удобно, так как, заранее не известны характеристики аппаратной платформы, на которой будет разворачиваться информационная система. Конфигурацию разделов, в будущем, будет достаточно просто изменить.

После того, как процесс установки завершен, необходимо выполнить действия, направленные на оптимизацию установленной операционной системы. Все действия производятся в командной строке от имени суперпользователя (*root*). Приглашение командной строки, начинающееся с символа "#", означает выполнение команд от имени суперпользователя:

`# vim /etc/sysconfig/selinux` - выставить значение параметра `selinux=disabled`. SELinux (*Security-Enhanced Linux* – Linux с улучшенной безопасностью) – реализация системы принудительного контроля доступа, которая может работать параллельно с классической дискреционной системой доступа. Эта система входит в стандартное ядро Linux, для ее работы требуется определять и задавать политики, для выполняющихся процессов в системе. Настройка SELinux довольно сложный процесс, который имеет смысл проводить на стабильно функционирующем сервере. Рекомендуется отключить эту систему.

`# vim /etc/sysconfig/network-scripts/ifcfg-eth0`, - так как, при конфигурировании виртуальной машины был выбран режим NAT (*Network Address Translation* – преобразование сетевых адресов), для присвоения, встроенным DHCP-сервером менеджера виртуальных машин, сетевого адреса гостевой ОС, необходимо в вышепредставленный файл записать три параметра:

```
DEVICE=eth0
BOOTPROTO=dhcp
ONBOOT=yes
```

`# chkconfig network on`, - включить запуск сетевого сервиса при загрузке ОС

`# service network start`, - запустить сетевой сервис

После выполнения последней команды, системой будет получен сетевой адрес (например, 192.168.220.168), который будет использоваться далее в этой работе.

```
# yum localinstall --nogpgcheck
http://download1.rpmfusion.org/free/fedora/rpmfusion-free-release-
stable.noarch.rpm http://download1.rpmfusion.org/nonfree/fedora/rpmfusion-
nonfree-release-stable.noarch.rpm
```
, - добавление дополнительных репозиториев

```
# yum update –y
```
, - обновить все пакеты операционной системы

```
# reboot
```
, - после обновления ОС, необходимо перезагрузиться

```
# yum install vim mc elinks mlocate unzip perl cpan gcc gcc-c++ gfortran
python-devel python-setuptools man httpd lighttpd lighttpd-fastcgi libpqxx
libpqxx-devel postgresql postgresql-server postgresql-devel proj proj-epsg
proj-devel postgis java-1.6.0-openjdk java-1.6.0-openjdk-devel httperf
```
, - так как, дистрибутив ОС устанавливался с минимальным количеством пакетов, нужно добавить необходимые для дальнейшей работы пакеты, вместе с зависимостями

```
# chkconfig --list | grep вкл
```
, - с помощью данной команды можно узнать, какие системные сервисы стартуют при запуске системы

Для уменьшения количества потребляемых ресурсов ОС, отключим не востребованные системные сервисы:

```
# chkconfig ip6tables off
# chkconfig iptables off
# chkconfig netfs off
# chkconfig sendmail off
# chkconfig udev-post off
# reboot
```

На рисунке 2.7 представлен вывод утилиты `free, который показывает` потребление операционной системой оперативной памяти.

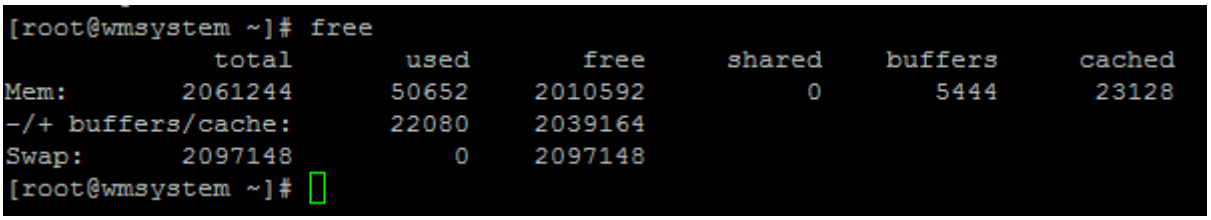


Рисунок 2.7 – Вывод утилиты free

Из представленной выше информации видно, что потребление памяти ОС находится в пределах 60Мб, при общем объеме виртуальной памяти 4Гб.

Отдельно, рассмотрим процесс настройки и работы с PostGIS/PostgreSQL. Необходимые исходные данные в формате Shapefile уже определены, требуется конвертировать их в SQL-дамп для PostGIS. Это можно сделать с помощью скрипта командной оболочки Linux, приведенного в предыдущем разделе:

```
# chmod +x convert.sh
# cd poppnt/
# ./convert.sh
```

После завершения процесса конвертации, в текущем каталоге появится файл `poppnt.sql`, который позже необходимо будет применить к базе данных.

`# /etc/init.d/postgresql initdb`, - инициализировать базу данных postgres

`# /etc/init.d/postgresql start`, - запустить службу postgresql

`# chkconfig postgresql on`, - запускать службу при загрузке системы

`# su postgres`, - войти под пользователем postgres

`$ createdb –U postgres wmsystem`, - создать базу данных wmsystem

```
$ createlang plpgsql wmsystem
```

`$ psql -d wmsystem -f /usr/share/pgsql/contrib/postgis-1.5/postgis.sql`, - добавить возможность манипулирования географическими объектами

```
$ psql -d wmsystem -f /usr/share/pgsql/contrib/postgis-
1.5/spatial_ref_sys.sql
```

`$ exit`, - выйти из-под пользователя postgres

`# vim /var/lib/pgsql/data/pg_hba.conf`, - сменить метод аутентификации локальных клиентов на `trust`, для Unix-сокетов и соединений IPv4

`# vim /var/lib/pgsql/data/postgresql.conf`, - активировать опции `listen_addresses` и `port`, тем самым, дав возможность подключаться в PostgreSQL по сети

`# /etc/init.d/postgresql restart`, - перезапустить PostgreSQL после внесения изменений в настройки

`# psql -d wmsystem –U postgres -w –f poppnt/sql/poppnt.sql`, - добавить слой poppnt в базу данных wmsystem

Итак, тестовая таблица с данными PostGIS успешно создана, разрешены подключения к базе данных по локальной сети. На этом, первоначальная настройка выбранного дистрибутива ОС закончена, и необходимо создать снимок ее состояния, что бы впоследствии, можно было вернуть ОС к этому состоянию.

2.3 Определение условий тестирования

При проведении сравнительного анализа, требуется измерить скорость представления результата, картографическим сервером, для каждого выбранного слоя исходных пространственных данных. Сравниваемые программные продукты поддерживают многопоточность, поэтому оценка будет проводиться для нескольких конфигураций виртуальной машины: с использованием одного и двух ядер CPU. Изменить количество доступных ядер для виртуальной машины, можно в ее настройках, которые представлены на рисунке 2.8.

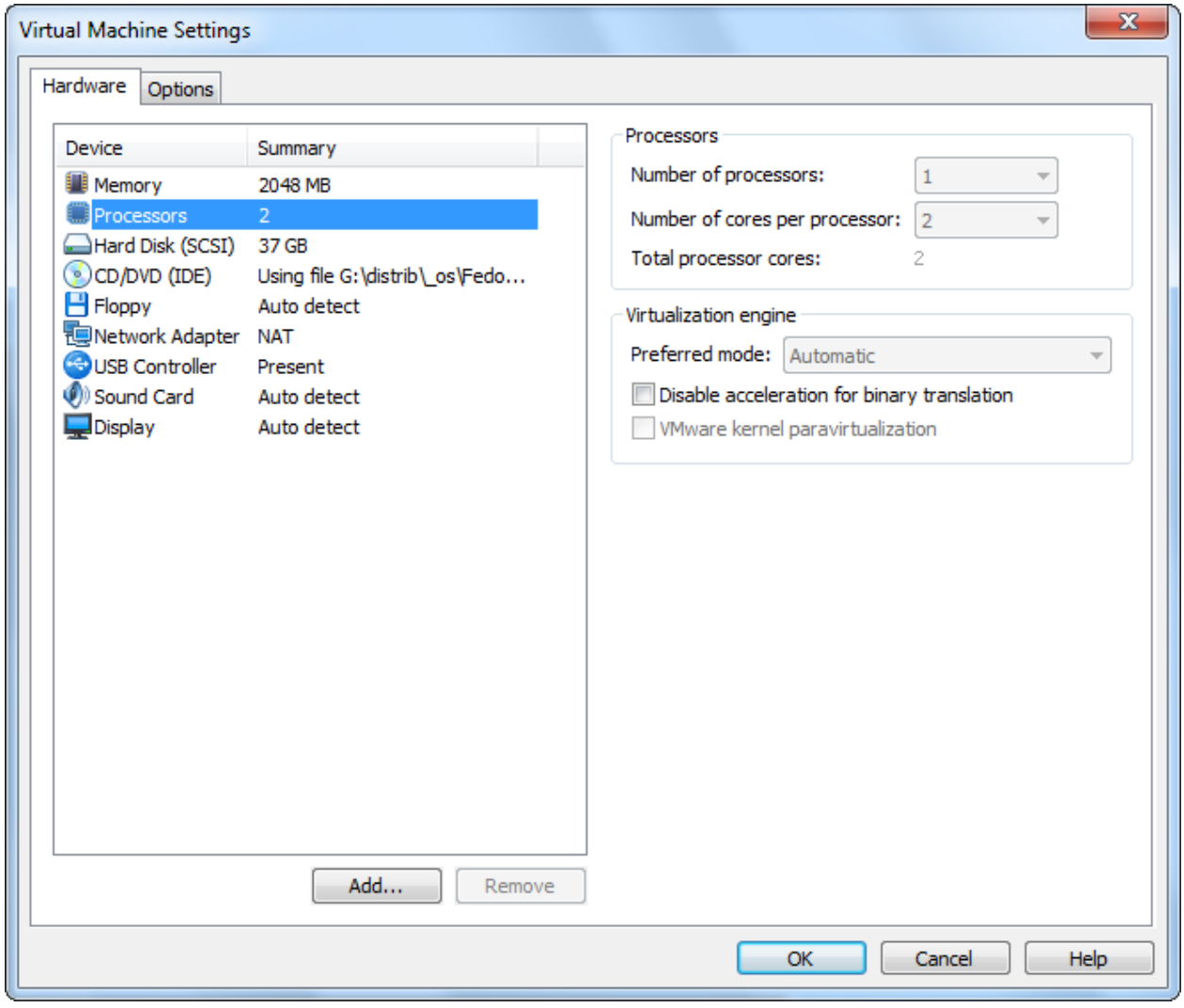


Рисунок 2.8 – Настройки виртуальной машины

Оценка производительности будет осуществляться для следующих наборов параметров:

- Конфигурация с одним ядром CPU
    - Слой “Сетка” (Shapefile, crdlin); проекция EPSG:4326; охват (-180.000000, 40.000000) - (180.000000, 84.000000); формат вывода PNG; размер изображения 900x100 пикселей
    - Слой “Реки” (Shapefile, hdrlin); проекция EPSG:4326; охват (-180.000000, 40.000000) - (180.000000, 84.000000); формат вывода PNG; размер изображения 900x100 пикселей
    - Слой “Пункты” (PostGIS, poppnt); проекция EPSG:4326; охват (-180.000000, 40.000000) - (180.000000, 84.000000); формат вывода PNG; размер изображения 900x100 пикселей
    - Слой “Сетка” (Shapefile, crdlin); проекция EPSG:3576; охват (-5133549.567108, -5397733.446068) - (5397733.446068, 1667991.365941); формат вывода PNG; размер изображения 800x600 пикселей
    - Слой “Реки” (Shapefile, hdrlin); проекция EPSG:3576; охват (-5133549.567108, -5397733.446068) - (5397733.446068, 1667991.365941); формат вывода PNG; размер изображения 800x600 пикселей
    - Слой “Пункты” (PostGIS, poppnt); проекция EPSG:3576; охват (-5133549.567108, -5397733.446068) - (5397733.446068, 1667991.365941); формат вывода PNG; размер изображения 800x600 пикселей
- Конфигурация с двумя используемыми ядрами CPU
    - Слой “Сетка” (Shapefile, crdlin); проекция EPSG:4326; охват (-180.000000, 40.000000) - (180.000000, 84.000000); формат вывода PNG; размер изображения 900x100 пикселей
    - Слой “Реки” (Shapefile, hdrlin); проекция EPSG:4326; охват (-180.000000, 40.000000) - (180.000000, 84.000000); формат вывода PNG; размер изображения 900x100 пикселей
    - Слой “Пункты” (PostGIS, poppnt); проекция EPSG:4326; охват (-180.000000, 40.000000) - (180.000000, 84.000000); формат вывода PNG; размер изображения 900x100 пикселей

- Слой “Сетка” (Shapefile, crdlin); проекция EPSG:3576; охват (-5133549.567108, -5397733.446068) - (5397733.446068, 1667991.365941); формат вывода PNG; размер изображения 800x600 пикселей
- Слой “Реки” (Shapefile, hdrlin); проекция EPSG:3576; охват (-5133549.567108, -5397733.446068) - (5397733.446068, 1667991.365941); формат вывода PNG; размер изображения 800x600 пикселей
- Слой “Пункты” (PostGIS, poppnt); проекция EPSG:3576; охват (-5133549.567108, -5397733.446068) - (5397733.446068, 1667991.365941); формат вывода PNG; размер изображения 800x600 пикселей

Действия по просмотру результатов и анализа их скоростных характеристик, будут проводиться в хостовой ОС. Это возможно потому, что сетевой интерфейс виртуальной машины, фактически, является локальным интерфейсом хост компьютера. Значит, задержки в сетевой среде будут минимальными и ими можно пренебречь.

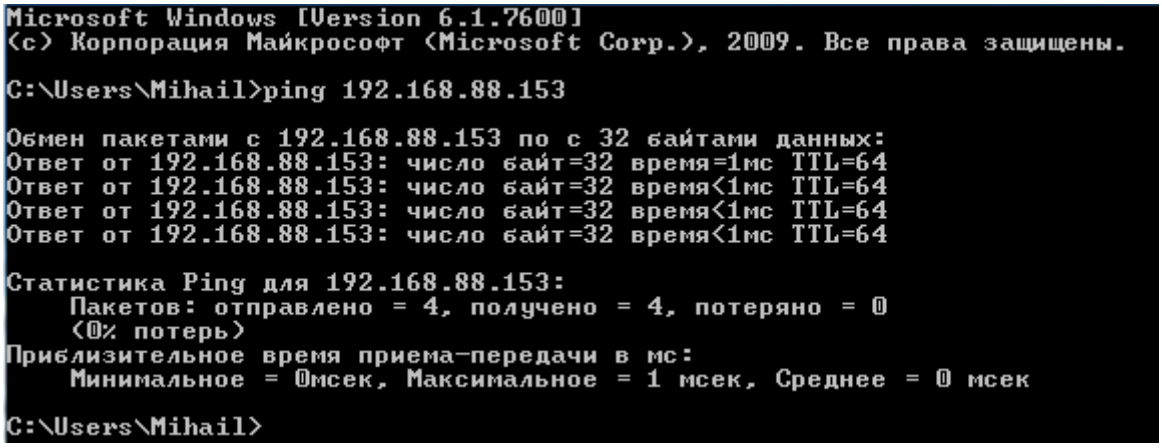


Рисунок 2.9 – Задержки в сетевой среде

Для визуальной оценки правильности настройки и работоспособности картографического wms-сервиса, в качестве wms-клиента можно использовать программное обеспечение QGIS. QGIS (*Quantum GIS*) – свободное программное обеспечение, для просмотра, редактирования и анализа геоданных.

Для оценки скоростных характеристик будет использоваться инструмент AB (*Apache Bench*). Данная утилита входит в стандартный дистрибутив Apache HTTP Server, и предназначена для нагрузочного тестирования web-серверов. Суть такого тестирования в том, что бы посылать на сервер определенное количество http-запросов, замеряя при этом время их обработки. Работа с данным инструментом происходит через командную строку. Определим достаточные для проведения исследования параметры, с которыми будет вызываться AB:

- *Количество одновременно посылаемых обращений.* Для проведения исследования, условимся посылать по одному одновременному обращению на ядро. Для исследуемого сервиса важно определить именно скорость отрисовки картографического изображения с необходимыми параметрами (время отдачи изображения будет пренебрежимо малым). Задавать же несколько одновременных подключений, эмулирующих одновременную работу нескольких пользователей, имеет смысл на этапе разработки сервиса.

- *Общее количество запросов.* Для проведения исследования, условимся посылать 5 (6 для конфигурации с двумя используемыми ядрами) запросов к web-сервису. Этого количества запросов хватит, что бы получить усредненное значение времени обработки запросов web-сервисом, игнорируя, возможно присутствующие, внутренние механизмы кэширования соединения с источниками данных и прочие.

Итак, вызов утилиты, для конфигурации с одним используемым ядром, будет осуществляться со следующими параметрами:

`ab –c1 –n5 "http://wms"`, - где `http://wms` – запрос к WMS-серверу

Для конфигурации с двумя используемыми ядрами, вызов утилиты будет выглядеть следующим образом:

`ab –c2 –n6 "http://wms"`, - где http://wms – запрос к WMS-серверу

Вывод утилиты достаточно информативен, поэтому, по ходу исследования, приводиться не будет. В качестве параметра для оценки будет использоваться среднее время в миллисекундах, затрачиваемых на один запрос, без учета конкурентных запросов (второй параметр *Time per request*).

2.4 Подготовка к исследованию картографического сервера GeoServer

2.4.1 Установка и запуск

Для проведения исследования будет использована последняя версия GeoServer (*geoserver-2.1-RC1-bin.zip*), которую можно загрузить на официальном сайте проекта. Программа распространяется в нескольких вариантах: бинарный платформонезависимый (именно этот вариант используется), web-архив, инсталлятор для Windows, инсталлятор для Mac OS X, и исходные коды. Используемый вариант требует наличия реализации виртуальной машины *Java*

Runtime Environment, которую в виде нескольких пакетов с зависимостями мы предусмотрительно установили при первоначальной настройке ОС.

Для выполнения дальнейших действий, необходимо определить несколько переменных окружения и осуществить запуск сервера:

```
# export JAVA_HOME=/usr/lib/jvm/jre/
# cd /srv, - дистрибутив с GeoServer будет располагаться в этом каталоге
# unzip geoserver-2.1-RC1-bin.zip
# cd geoserver-2.1-RC1
# ./startup.sh
```

После выполнения последней команды будет произведен запуск контейнера сервлетов, написанного на Java - *Jetty*, который используется в GeoServer по умолчанию. Он может использоваться как HTTP-сервер или в паре со специализированным HTTP-сервером (к примеру, с Apache HTTP Server). Что бы уменьшить влияние посторонних компонентов на результаты исследования, Jetty будет использоваться как самостоятельный HTTP-сервер, работающий на порту 8080.

Одним из достоинств GeoServer является наличие у него собственного web-интерфейса, что значительно облегчает его настройку, добавление источников пространственных данных и управление ими. При этом сохранятся возможность настройки посредством XML-файлов. Описывать работу с web-интерфейсом GeoServer в данной работе не имеет смысла, вся исчерпывающая информация по работе с ним представлена на официальном сайте проекта.

После выполнения всех действий, web-интерфейс GeoServer станет доступен по адресу http://192.168.220.168:8080/geoserver/web/.

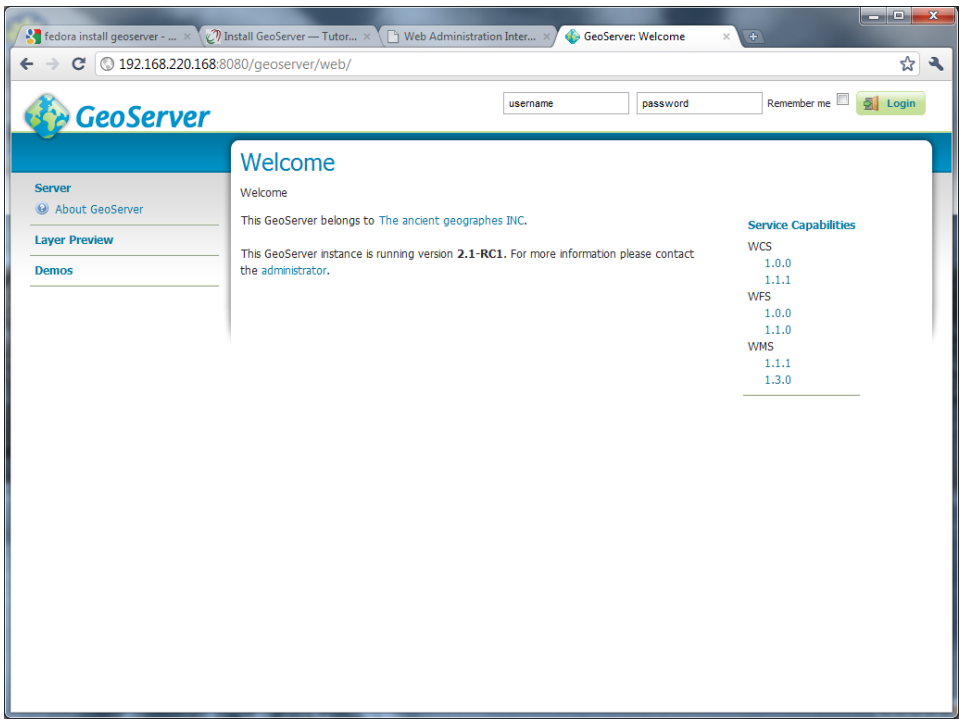


Рисунок 2.10 – Web-интерфейс GeoServer

2.4.2 Настройка и просмотр результата

В первую очередь, необходимо создать каталог, в котором будут храниться исходные пространственные данные, и загрузить их туда:

```
# mkdir /srv/geoserver-2.1-RC1/data_dir/data/wmsystem
```

Следующим шагом будет создание хранилища данных. Для этого, в меню слева, web-интерфейса, нужно перейти по ссылкам: *Workspaces/Add new workspace*. В поле *Name* и *Namespace URI* следует ввести *wmsystem*. Так же, можно присвоить только что созданному рабочему окружению атрибут “по умолчанию”, установкой галочки *Default Workspace*. Для сохранения нажать кнопку *Save*.

Теперь, необходимо указать местоположение источников данных, для Shapefile и для PostGIS:

- *Shapefile*. В меню слева перейти по ссылкам: *Stores/Add new Store/Shapefile*. Необходимо указать в *Data Source Name* имя загружаемого слоя (*crdlin* и *hdrlin*), и соответственно этому, выбрать источник данных. Необходимо определить кодировку слоя в UTF-8, с помощью списка *DBF charset*. Нажать кнопку *Save*.

- *Postgis*. В меню слева перейти по ссылкам: *Stores/Add new Store/PostGIS*. Необходимо указать в *Data Source Name* имя загружаемого слоя - *poppnt*. В секции настроек *Connection Parameters*, указать следующее: *host – localhost*; *database – wmsystem*; *user – postres*. Нажать кнопку *Save*.

Параметры слоя в GeoServer, можно задать, перейдя в меню слева по ссылкам *Layers/Add a new resource*. Далее необходимо выбрать одно из ранее созданных хранилищ данных в списке *Add layer from*, и перейти по ссылке *Publish*. Перейдя по этой ссылке, станут доступны параметры отображения слоя, такие как охват и проекция, в нужные поля необходимо ввести необходимые значения. Нажать кнопку *Save*. Стилистика слоя на данном этапе задаваться не будет.

Необходимо настроить три, ранее определенных, слоя данных. После настройки, визуальную оценку можно провести с помощью wms-клиента. Адрес wms-сервера, без передаваемых параметров, будет выглядеть следующим образом: *http://192.168.220.168:8080/geoserver/ows?SERVICE=WMS&*. Результат настройки GeoServer представлен на рисунке 2.11.

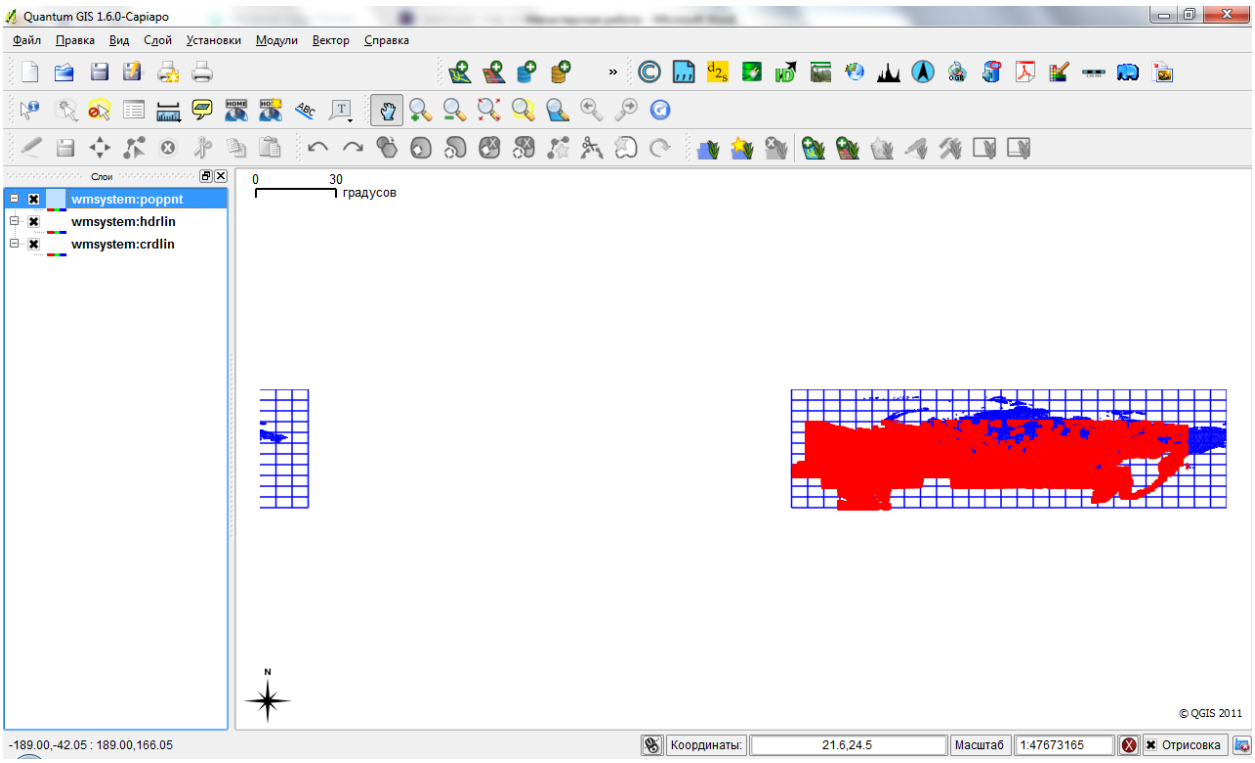


Рисунок 2.11 – Проверка работоспособности GeoServer

После проведения визуальной оценки работоспособности, можно приступить к измерению скорости представления данных.

2.5 Подготовка к исследованию картографического сервера MapServer

2.5.1 Установка и запуск

Установка MapServer была проведена на этапе первоначальной настройки дистрибутива ОС. Заключалась она в виде установки пакета *mapserver-5.6.3-3.fc14.i686*, и необходимых для него зависимостей. Основу MapServer составляет *CGI-программа mapserv* (путь к ней выглядит следующим образом: */usr/sbin/mapserv*), которая принимает от пользователя параметры (файл описания карты, слои карты, размер текущей карты и т.д.), после чего отображает запрашиваемую карту. Обычно, это происходит следующим образом: получив от пользователя запрос, MapServer генерирует растровый файл и, если указано, встраивает его в html-документ, отсылаемый пользователю. Какие слои будут участвовать при генерации файла, как именно они будут отображаться, будут ли подписаны объекты на карте, а так же многое другое указывается в специальном файле с расширением .map, который передается программе mapserv в качестве одного из параметров, заданных в запросе.

2.5.2 Настройка и просмотр результата

Прежде чем перейти непосредственно к настройке, необходимо убедиться в том, что установленная версия поддерживает стандарт WMS:

```
[root@wmsystem ~]# mapserv -v
MapServer version 5.6.3 OUTPUT=GIF OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP OUTPUT=SVG
 SUPPORTS=PROJ SUPPORTS=AGG SUPPORTS=CAIRO SUPPORTS=FREETYPE SUPPORTS=ICONV SUPP
ORTS=FRIBIDI SUPPORTS=WMS_SERVER SUPPORTS=WMS_CLIENT SUPPORTS=WFS_SERVER SUPPORT
S=WFS_CLIENT SUPPORTS=WCS_SERVER SUPPORTS=SOS_SERVER SUPPORTS=FASTCGI SUPPORTS=T
HREADS SUPPORTS=GEOS SUPPORTS=RGBA_PNG INPUT=TIFF INPUT=POSTGIS INPUT=OGR INPUT=
GDAL INPUT=MYGIS INPUT=SHAPEFILE
[root@wmsystem ~]#
```

Рисунок 2.12 – Проверка поддержки MapServer стандарта WMS

Из представленной выше информации, видно, с поддержкой каких параметров был собран MapServer. Параметр `SUPPORTS=WMS_SERVER` подтверждает поддержку WMS.

Так как mapserv является CGI-программой, для выполнения запросов к ней и получения необходимых ответов требуется HTTP-сервер. MapServer может работать в нескольких режимах: *CGI* (*Common Gateway Interface*) и *FastCGI*. Разработчики этого картографического сервера, рекомендуют использовать свой продукт в режиме FastCGI, для достижения более высокого

уровня производительности. Так же, поведение этого режима, более похоже на поведение встроенного HTTP-сервера GeoServer, чем CGI.

FastCGI – клиент-серверный протокол взаимодействия web-сервера и приложения, дальнейшее развитие технологии CGI. По сравнению с CGI является более производительным и безопасным. FastCGI ликвидирует множество ограничений CGI-программ. Проблема CGI-программ в том, что они должны быть перезапущены web-сервером при каждом запросе, что приводит к понижению производительности. FastCGI убирает это ограничение, сохраняя процесс запущенным и передавая запросы этому постоянно запущенному процессу. Это позволяет не тратить время на запуск новых процессов. В то время как CGI программы взаимодействуют с сервером через *STDIN* и *STDOUT* запущенного CGI-процесса, FastCGI-процессы используют *Unix Domain Sockets* (сокет межпроцессного взаимодействия) или TCP/IP для связи с сервером. Это дает следующее преимущество над обычными CGI-программами: FastCGI-программы могут быть запущенны где угодно в сети. Возможна обработка запросов, несколькими FastCGI-процессами, работающими параллельно.

Для запуска FastCGI процесса будет использован *Lighttpd*. Lighttpd – web-сервер, разрабатываемый с расчетом на быстроту и защищенность, а также соответствие стандартам. Это свободное программное обеспечение, распространяемое по лицензии BSD, доступное в версиях для Linux и других Unix-подобных операционных систем, а так же для Microsoft Windows. Проект Lighttpd начался со стремления автора реализовать web-сервер, который мог бы выдержать одновременно 10 тысяч соединений. Lighttpd использует, так называемую, асинхронную обработку сетевых соединений. Благодаря этому, загруженность сервера (в отличие от Apache) при доступе к файлам на диске не зависит от количества текущих соединений. В Lighttpd возможно использование особых системных вызовов для повышения производительности при передаче файлов. При этом задействуются не стандартные системные интерфейсы, а специфичные для платформы вызовы ядра операционной системы, и смена контекста CPU сводится к минимуму. Зачастую Lighttpd (как и имеющий схожую структуру nginx, но более сложный в настройке) используется для отдачи статического содержимого, в то время как его генерацией занимается более ресурсоемкое приложение. Данный факт, как нельзя лучше подходит для организации взаимодействия с MapServer.

Для настройки Lighttpd в режиме FastCGI для MapServer, необходимо внести следующие изменения в конфигурационный файл:

- В списке загружаемых модулей `server.modules`, необходимо активировать значение `mod_fastcgi`

- Задать значение опции `server.port = 8080`
- Добавить следующие строки, конфигурирующие `mod_fastcgi`:

```
fastcgi.server = ( "/maps" =>
                                    ( "localhost" =>
                        (
                          "socket" => "/var/run/lighttpd/mapserver-
fastcgi.socket",
                          "check-local" => "disable",
                          "bin-path" => "/usr/sbin/mapserv",
                          "min-procs" => 1,
                          "max-procs" => 6,
                          "max-load-per-proc" => 4,
                          "idle-timeout" => 20
                          )
                          )
   )
```

Детальное описание опций конфигурационного файла Lighttpd, можно найти в документации, представленной на официальном сайте проекта. Запуск web-сервера можно произвести следующим образом:

`# /etc/init.d/lighttpd start`

`# chkconfig lighttpd on`

`# ps aux | grep mapserv`, - с помощью этой команды можно убедиться в запуске fastcgi-процессов

Если все действия выполнены правильно, то перейдя в браузере по адресу *http://192.168.220.168:8080/maps*, можно увидеть следующую строку:

No query information to decode. QUERY_STRING is set, but empty.

Данная строка означает, что MapServer успешно запущен, и ему можно передавать необходимые параметры.

Для выполнения дальнейших действий, необходимо произвести следующие настройки:

`# mkdir /srv/wmsystem`, - начальный каталог

`# mkdir /srv/wmsystem/maps`, - здесь будут храниться файлы, относящиеся к карте

`# mkdir /srv/wmsystem/maps/data`, - сюда необходимо загрузить исходные данные в формате Shapefile

`# toucn /srv/wmsystem/maps/wms.map`, - map-файл, в который заносятся все настройки карты

`# vim /usr/share/proj/epsg`, - в данный файл необходимо добавить описание проекции EPSG:3576:

```
<3576> +proj=laea +lat_0=90 +lon_0=90 +x_0=0 +y_0=0 +ellps=WGS84 +datum=WGS84 +units=m +no_defs
```

`# chmod 0777 /var/www/lighttpd/wmsystem/tmp/`, - временно, разрешим всем пользователям запись в данный каталог

Как уже отмечалось выше, все настройки, относящиеся к карте и ее отображению, MapServer хранит в так называемых map-файлах. Выше, мы создали такой файл. Его содержание (с пояснениями), для организации WMS сервиса и проведения исследования, выглядит следующим образом:

```
MAP # ключевое слово, которое обзначает начало "map" объекта
    NAME        "WMSYSTEM_WMS"
    STATUS      ON
    IMAGETYPE   PNG
    EXTENT      -180.00 40.00 180.00 84.00 # охват карты
    SIZE        900 100 # размер карты в пикселях
    SHAPEPATH   "data/" # местоположение shp-файлов
    UNITS       DD # используемые единицы измерения, lat/lon
    IMAGECOLOR  255 255 255 # цвет фона карты в формате RGB
    CONFIG      "PROJ_LIB" "/usr/share/proj" # путь к базе данных EPSG кодов PROJ.4
    CONFIG      "MS_ERRORFILE" "/tmp/ms_debugs.log" # путь к лог-файлу
    DEBUG       5 # уровень логирования, показывать все ошибки

    WEB
        TEMPLATE "templates/template.html"
        IMAGEPATH "/var/www/lighttpd/wmsystem/tmp/"
```

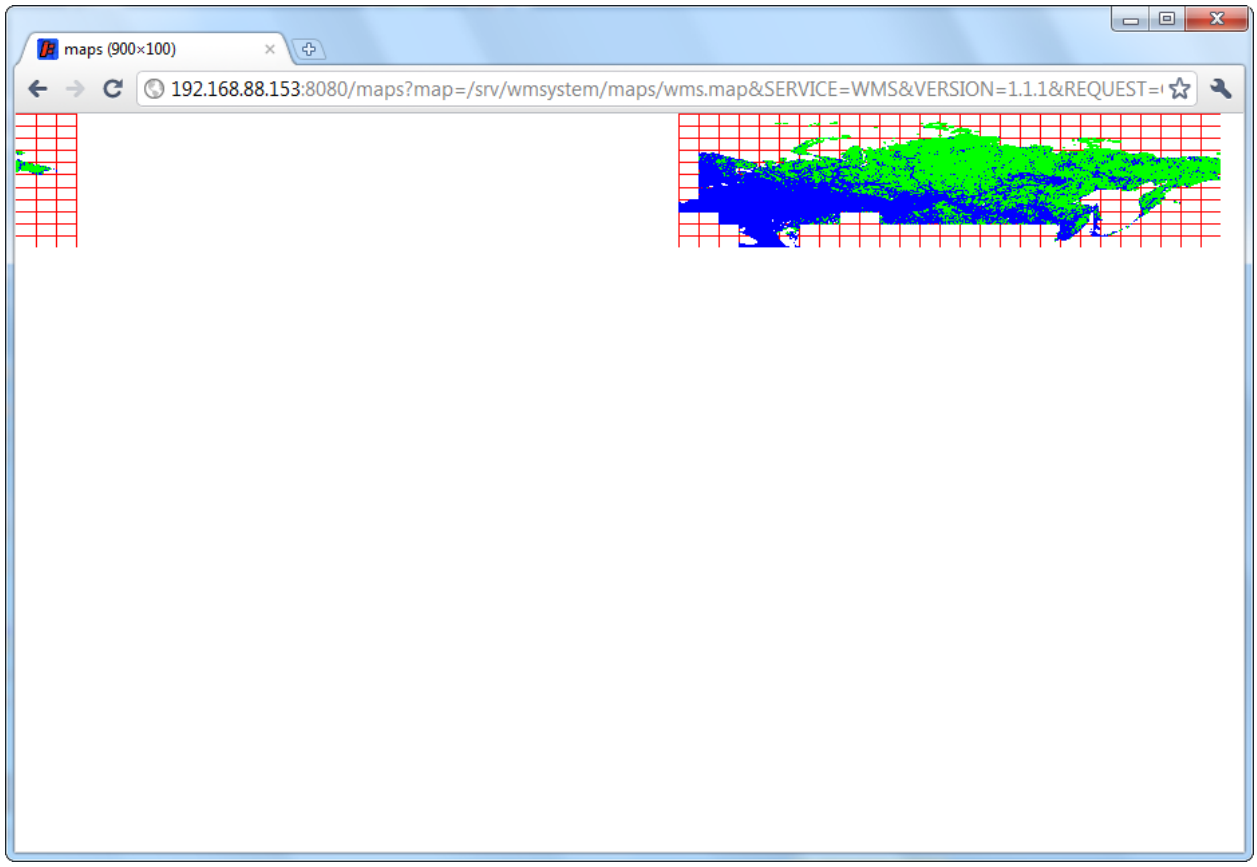
```
        IMAGEURL "/tmp/"
        METADATA
            mapfile_encoding "UTF-8"
            wms_title "WMSystem WMS"
            wms_abstract "This is the WMS server from WMSystem"
            wms_onlineresource
"http://localhost:8080/maps?map=/srv/maps/wms.map&"
            wms_srs "EPSG:4326 EPSG:3576"
            wms_getfeatureinfo
"http://localhost:8080/maps?map=/srv/maps/wms.map&"
            wms_featureinfoformat "text/plain"
            wms_feature_info_mime_type "text/html"
        END
    END

    PROJECTION
        "init=epsg:4326" # исходная проекция карты
    END
  OUTPUTFORMAT
        NAME aggpng24 # задействуем возможности сглаживания
        DRIVER AGG/PNG
        MIMETYPE "image/png"
        IMAGEMODE RGB
        EXTENSION "png"
    END

    LAYER
        NAME crdlin # название слоя
        DATA crdlin # название shp-файла
        STATUS ON # переключатель слоя на карте
        TYPE LINE # тип слоя
        TEMPLATE "void" # указатель шаблона, используется пустой шаблон
        PROJECTION
```

```
            "init=epsg:4326" # исходная проекция слоя
        END
        METADATA
            wms_title "Сетка"
            wms_abstract "Сетка описание"
            wms_srs "EPSG:4326 EPSG:3576" # в каких проекциях будет доступен
слой
            wms_include_items "all" # включить все элементы в результат при
выборке
        END
        CLASS
            NAME "crdlin"
            STYLE
                COLOR 255 0 0 # цвет линии
            END
        END
    END

    LAYER
        NAME hdrlin
        DATA hdrlin
        STATUS ON
        TYPE LINE
        TEMPLATE "void"
        PROJECTION
            "init=epsg:4326"
        END
        METADATA
            wms_title "Реки"
            wms_abstract "Реки описание"
            wms_srs "EPSG:4326 EPSG:3576"
            wms_include_items "all"
        END
```

```
        CLASS
            NAME "crdlin"
            STYLE
                COLOR 0 255 0
            END
        END
    END


    LAYER
        NAME poppnt
        CONNECTIONTYPE postgis # тип соединения
        CONNECTION "user=postgres dbname=wmsystem host=127.0.0.1" #
параметры, необходимые для подключения к БД
        DATA "the_geom from poppnt" # таблица с пространственными данными
        STATUS ON
        TYPE POINT
        TEMPLATE "void"
        PROJECTION
            "init=epsg:4326"
        END
        METADATA
            wms_title "Пункты"
            wms_abstract "Пункты описание"
            wms_srs "EPSG:4326 EPSG:3576"
            wms_include_items "all"
        END
        CLASS
            NAME "poppnt"
            STYLE
                COLOR 0 0 255
            END
        END
    END
```

```
END
```

Результат можно увидеть, выполнив в браузере запрос к адресу: *http://192.168.88.153:8080/maps?map=/srv/wmsystem/maps/wms.map&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&LAYERS=crdlin,hdrlin,poppnt&SRS=EPSG:4326&BBOX=-180,40,180,84&WIDTH=900&HEIGHT=100&FORMAT=image/png*



2.13 – Проверка работоспособности MapServer

На этом минимально необходимая настройка MapServer закончена, и можно переходить к оценке его производительности.

2.6 Исследование

Результаты измерений утилитой *ab*, для заданных наборов параметров, представлены в таблицах 2.1 и 2.2.

Таблица 2.1 – Результаты измерений производительности картографических серверов, для конфигурации с одним ядром CPU

Параметры	GeoServer	MapServer
EPSG:4326: Географическая сетка	293,61	43,73
EPSG:4326: Водные объекты	5470,45	10895,60
EPSG:4326: Населенные пункты	12177,74	1187,14
EPSG:4326: Общий	17217,58	13630,27
EPSG:3576: Географическая сетка	563,11	166,46
EPSG:3576: Водные объекты	30000,00	19192,78
EPSG:3576: Населенные пункты	19362,04	1704,30
EPSG:3576: Общий	30000,00	20898,65

Таблица 2.2 – Результаты измерений производительности картографических серверов, для конфигурации с двумя ядрами CPU

Параметры	GeoServer	MapServer
EPSG:4326: Географическая сетка	257,16	14,09
EPSG:4326: Водные объекты	3456,49	3698,28
EPSG:4326: Населенные пункты	7957,58	701,91
EPSG:4326: Общий	10050,69	3544,65
EPSG:3576: Географическая сетка	477,46	30,40
EPSG:3576: Водные объекты	30000,00	6925,82
EPSG:3576: Населенные пункты	7425,10	620,28
EPSG:3576: Общий	30000,00	7971,53

Для полей, в результате которых стоит символ “*”, оценка производительности в числовом эквиваленте произведена не была, это связано с тем, что для этих запросов время ожидания сервера было слишком велико. Это не говорит о том, что картографический сервер не смог выполнить отрисовку изображения, это говорить лишь о том, что отрисовка выполнялась слишком долго. Примем для таких результатов значение времени отрисовки равное 30000 мс. На рисунках 2.14 и 2.15 приведены гистограммы результатов измерений производительности картографических серверов.

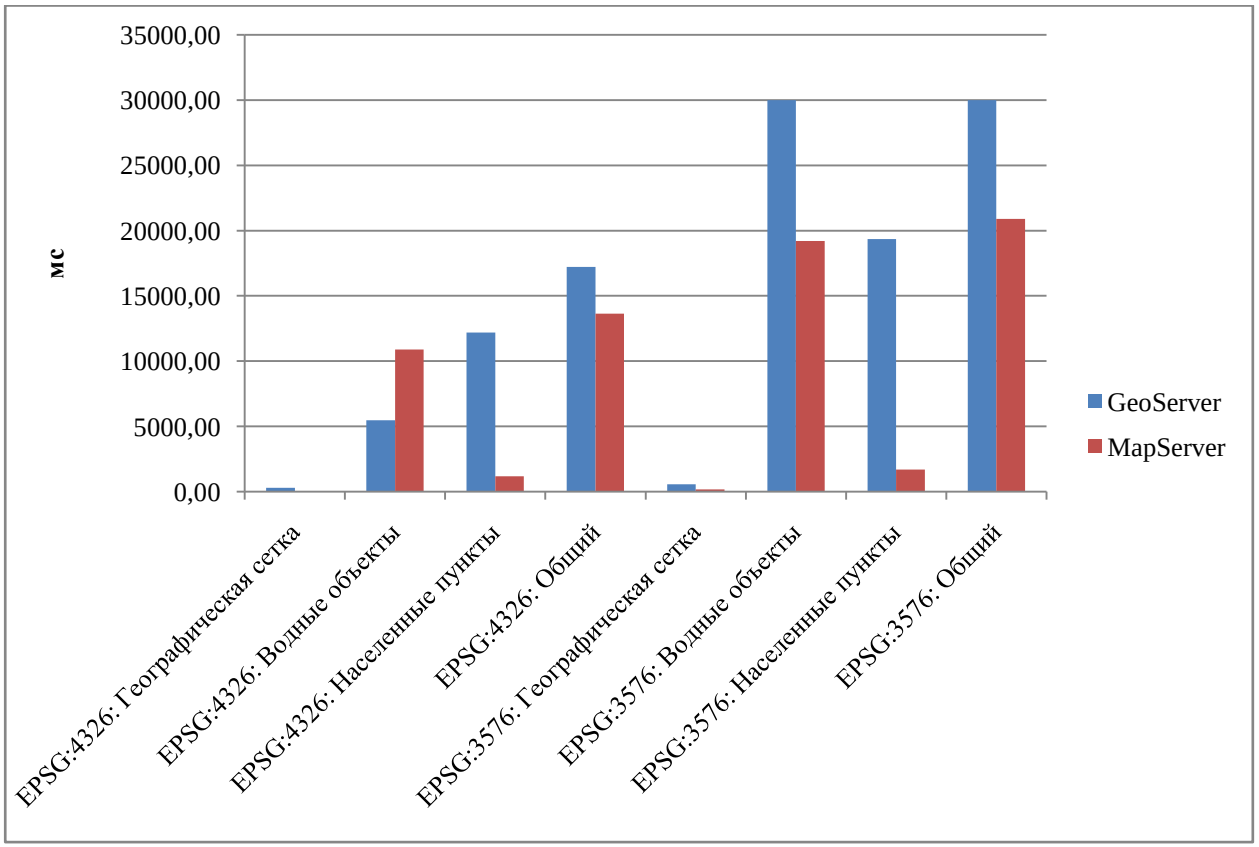


Рисунок 2.14 – Гистограмма результатов измерения производительности картографических серверов, для конфигурации с одним ядром CPU

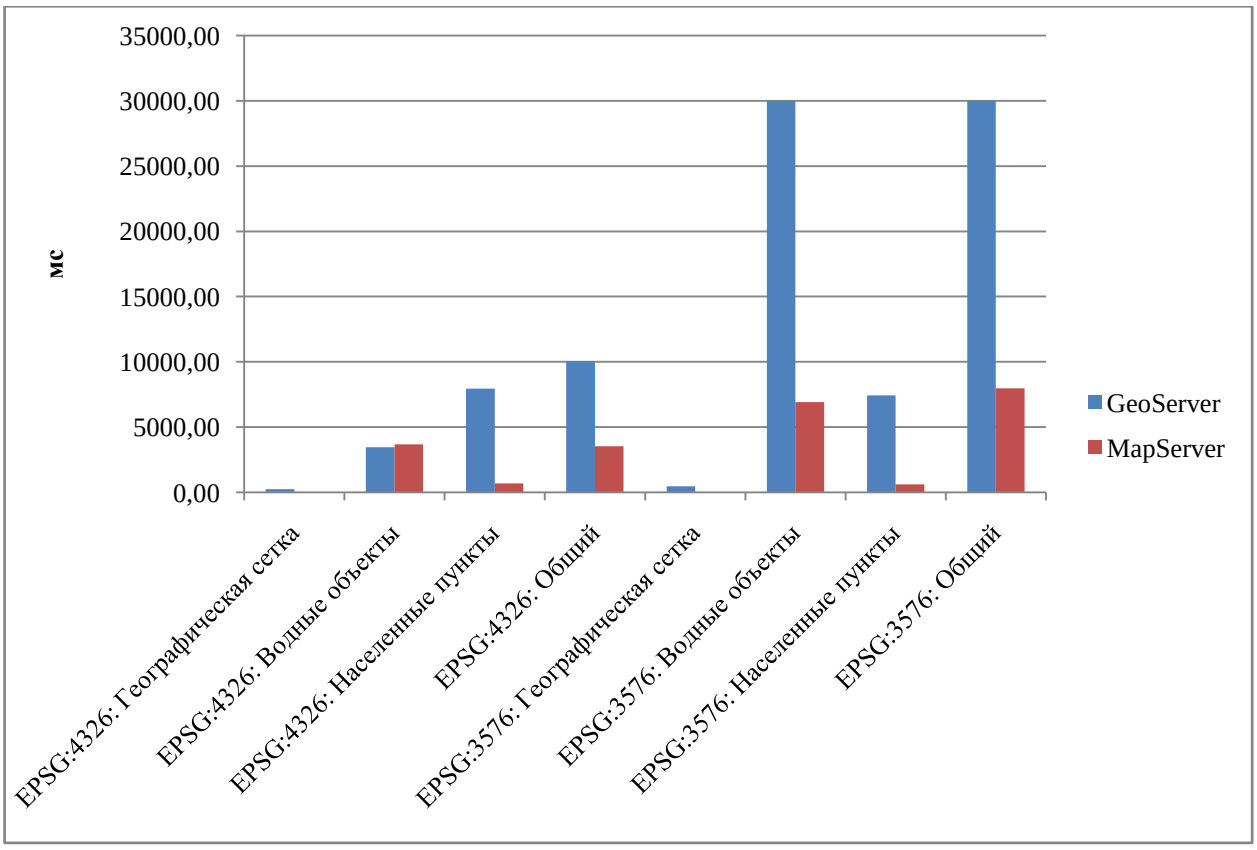


Рисунок 2.15 - Гистограмма результатов измерения производительности картографических серверов, для конфигурации с двумя ядрами CPU

Из представленных гистограмм прекрасно видно, что практически для всех наборов параметров, MapServer показывает наилучшие результаты. Исключение составляет лишь слой “Водные объекты”. Этот слой содержит большое количество ГИС-объектов, и с его отрисовкой лучше справился GeoServer, возможно, из-за особенностей внутренней реализации многопоточности. Но с задачами перепроецирования пространственных данных GeoServer справляется очень медленно, равно как и с задачей отображения данных, хранящихся в PostGIS. В общем случае, необходимость перепроецирования данных, увеличивает время отрисовки картографического изображения на 30-40%.

Прекрасные результаты можно получить, используя картографический сервер в конфигурации с многоядерными CPU. Время, затрачиваемое на отрисовку слоя с использованием нескольких ядер, в несколько раз меньше времени, затрачиваемого на отрисовку с использованием одного ядра CPU.

Использование MapServer - как раз тот случай, когда сложность настройки компенсируется высоким быстродействием. Построение решения задачи будет осуществляться на основе этого

картографического сервера, а разработка информационной системы производиться в конфигурации с использованием нескольких ядер CPU.

2.7 Сторонние исследования

Существует одно исследование [7], которое так же подтверждает полученные нами ранее результаты. Это исследование проводилось на конференции FOSS4G 2009 (*Free and Open Source Software for Geospatial*), как соревнование за звание “Самый быстрый сервер WMS”. Планировалось участие четырех претендентов, но разработчики таких лидеров рынка, как ESRI и ERDAS, в последний момент, по разным причинам, отказались участвовать в соревновании. Поэтому соревнование проводилось между двумя свободными продуктами: GeoServer и MapServer.

Коротко, опишем основные отличия исследования FOSS4G, от проводимого выше исследования:

- На мероприятии, проводимом FOSS4G, не использовалось сглаживание картографического изображения.
- Не использовалось перепроецирование данных.
- Использовалось несколько серверов большой мощности, с разделением функциональности.
- Использовались более старые версии библиотек и программ.
- Оценка производилась приложением JMeter с использованием большого числа конкурентных запросов и эмуляцией повседневной нагрузки.
- Генерировались изображения с разрешениями 640x480 и 1024x768 пикселей.
- Использовались растровые типы пространственных данных.

Победителем был признан MapServer. В операциях над векторными данными различия небольшие, на растровых данных MapServer опережает GeoServer в 1.5-2.5 раза. Причем, MapServer участвовал в двух вариантах – cgi и fastcgi, последний показывает стабильно 50-60% прироста скорости. Эти данные, подтверждают результаты исследования проводимого нами ранее.

2.8 Дополнительные условия

Создаваемая web-карта (как и web-картография в целом) обладает рядом особенностей, которые определяются средой их публикации [6]. Требования к любой карте, предназначенной для просмотра на экране монитора (электронной карте), существенно отличаются от правил подготовки карт для печати. Играют роль как ограничения средств отображения (разрешение, яркость, контраст монитора), так и ограничения по времени визуализации (сложность оформления и объем пространственных данных должны обеспечить разумное время прорисовки карты). Важное отличие заключается в том, что электронная карта, как правило, существует в нескольких масштабах – для удобства навигации. А учитывая то, что карта публикуется в виде web-сервиса, возникает ряд дополнительных условий, связанных с передачей данных по сети, визуализацией данных на сервере, а не на клиенте – и это так же замедляет работу.

Прежде чем приступать к построению решения задачи и непосредственно разработке, следует определиться с тем, что хотелось бы получить в результате. Часть необходимых условий и компонентов были определены в предыдущих разделах, а часть, предстоит определить на данном этапе.

Тип сервиса

Картографические web-сервисы принято делить на две большие группы: динамические и кэшируемые. В первом случае каждый раз при масштабировании или перемещении происходит обращение к данным и перерисовка изображения с учетом нового охвата. При этом клиент получает только готовые растровые картинки - их прорисовка осуществляется сервером.

В случае кэшированного сервиса мозаика растровых изображений (*тайлов*) для каждого масштаба готова заранее, поэтому работа выполняется на порядок быстрее. Следовательно, сложность оформления и самих данных никак не сказывается на скорости отображения – это большой плюс. Но имеются и очевидные ограничения: при изменении данных необходимо пересчитать кэш, нельзя редактировать объекты и управлять отображением слоев, список масштабов фиксирован. Также стоит помнить, что пересчет кэша может занять несколько суток и даже недель, в зависимости от того, насколько сложная карта лежит в его основе.

Выбор между динамическим и кэшируемым сервисом не столь очевиден, как может показаться на первый взгляд. Как правило, кэшируемый сервис имеет смысл делать при редко или фрагментарно обновляющихся данных, для создания карт-подложек под динамические сервисы, или если проект сам по себе небольшой – кэш можно быстро пересчитать. Существует еще

вариант многослойного кэша, когда каждый слой представляет собой отдельную растровую мозаику. Однако производительность сервиса в таком случае оставляет желать лучшего – ведь web-серверу приходится динамически накладывать растры слоев друг на друга. Такой тип кэша имеет очень ограниченное применение.

В целом, для разрабатываемой системы, наиболее преимущественно выглядит динамический web-сервис, так как он сможет в реальном времени отобразить изменения геофизических параметров на карте, если таковые появятся, а так же, даст возможность свободно переключаться между слоями. В дальнейшем, в качестве рекомендации, возможен вариант использования кэширующего сервиса, именно для подложки web-карты, и нанесение на нее слоев с динамическими данными.

Проекция карты и проекция данных

MapServer позволяет проецировать данные “на лету” (при визуализации), так что проекция карты может не совпадать с проекцией, в которой хранятся координаты объектов в источнике данных. Однако перепроецирование на лету не рекомендуется, поскольку, как мы убедились, оно существенно замедляет отображение карты и выполняется все же не идеально.

Отсюда следует правило: проекция данных должна совпадать с проекцией карты. При выборе проекции можно руководствоваться разными критериями – от эстетичности контуров (что немаловажно) и характера искажений до удобства и точности измерений. Удобный способ контроля проекции — использование наборов классов объектов в источнике пространственных данных. Наборы обычно организуют по тематике: транспорт, растительность и т.п. Внутри набора все классы должны иметь единую пространственную привязку – это важно для правил топологии. Таким образом, просто скопировать туда данные в другой проекции не удастся, сначала их придется перепроецировать. Если требуется совмещать несколько сервисов в одном web-приложении (накладывать карты друг на друга), то их проекция должна обязательно совпадать. Иначе часть сервисов попросту не будет отображаться.

В качестве основной проекции для отображения web-карты будет использоваться проекция *North Pole LAEA Russia* (EPSG:3576), а в качестве дополнительной – *географическая проекция* EPSG:4326. Для обеспечения совпадения основной проекции карты и проекции данных, данные должны быть перепроецированы.

Масштабный ряд

Это не менее важный аспект, чем выбор проекции. Кэшируемый картографический сервис имеет заранее определенную схему тайлов. Помимо параметров растра (формат, сжатие и т.п.) она включает и список масштабов. Очевидно, генерируя картографическое изображение заранее, мы не можем заложить в схему все возможные масштабы и тем самым обеспечить непрерывную навигацию. Иначе бы кэш строился бесконечно. Поэтому список масштабов всегда ограничен и зависит от охвата карты, подробности данных и многих других факторов. Обычно масштабный ряд состоит не более чем из 10-15 уровней.

Несмотря на то, что динамический сервис не привязан к определенным масштабам, задача их выбора от этого не становится менее актуальной. При увеличении или уменьшении изображения должна меняться подробность данных и, возможно, их состав. Соответствующим образом меняется и оформление. Предусмотреть уникальное поведение карты на любом масштабе невозможно, поэтому наличие предопределенного масштабного ряда окажется крайне полезным. Это позволит прорабатывать карту на “опорных” масштабах, не концентрируясь на том, что происходит между ними.

Необходимо заранее определиться с тем, какие масштабы будут использоваться, и в дальнейшем ориентироваться на них при генерализации и оформлении данных. Создаваемый web-сервис будет охватывать масштаб нашей страны, следовательно, для обеспечения просмотра карты пользователями с маленьким разрешением монитора (или портативного устройства), необходимо использовать минимальный масштаб 1:100000000. Исходя из этого, в проекте предполагается использование следующей масштабной линейки:

1. *2000,000000*
2. *5000,000000*
3. *10000,000000*
4. *25000,000000*
5. *50000,000000*
6. *100000,000000*
7. *200000,000000*
8. *300000,000000*
9. *500000,000000*
10. *1000000,000000*
11. *3000000,000000*
12. *10000000,000000*

13. *25000000,000000*
14. *35000000,000000*
15. *50000000,000000*
16. *100000000,000000*

В случае объединения нескольких кэшируемых сервисов в одном web-приложении желательно, чтобы их схемы совпадали.

Генерализация данных

Когда известен масштабный ряд, осмысленнее проходит и процесс генерализации данных. Можно использовать инструменты упрощения, сглаживания, отбора, объединения объектов и просматривать результаты в приложении ГИС, оценивая насколько их подробность, соответствует масштабам. Генерализация данных является обязательной. Более того, поскольку разрешающая способность монитора (~100 dpi) весьма ограничена по сравнению с возможностями принтера и бумаги (300 dpi и более), геометрическая подробность электронной карты должна быть соответственно меньше – минимум в 2-3 раза.

Каких-либо четких критериев оценки этой подробности нет. Эту операцию приходится производить интуитивно (“на глаз”). Скажем, если точки по границе контура расположены плотнее, чем через 0,5мм, вряд ли они будут различимы на экране. Опытный картограф всегда имеет в своем арсенале подобные визуальные критерии.

В результате формальной генерализации может нарушиться географическое правдоподобие (например, участок дороги окажется на территории соседнего государства). Поэтому необходимо по возможности использовать правила топологии и проверять данные как до, так и после генерализации. Визуальная оценка также обязательна.

Оформление данных

На самом деле, генерализация и оформление данных взаимозависимы. После того, как применяется определенный стиль отображения, допустим, к дорогам, становится ясно, что при данной толщине линии подробность избыточна. Так что можно говорить о том, что процесс генерализация-оформление итеративен: последовательными приближениями можно достигнуть удовлетворительного результата. Но в целом генерализация предшествует оформлению. Повторное упрощение может и не понадобиться.

При выборе условных обозначений каждый масштаб требует индивидуального подхода. Однако масштабный ряд в визуальном отношении представляет собой плавную последовательность: соседние масштабы, как правило, не должны иметь резких различий в плане

цветовой гаммы, стилей и т.п. Иначе при навигации у пользователей будут складываться некомфортные ощущения. Все-таки, это одна карта, но она оптимизирована для просмотра на разных масштабах.

При выборе стилей отображения для объектов и подписей нужно принимать в расчет три основных фактора: эстетичность, читаемость, производительность (генерализация также непосредственным образом влияет на эти три компоненты). Тема семантического соответствия знаков и объектов слишком обширна, и она не будет рассматриваться в данной работе. В случае динамического сервиса почти всегда чем-то приходится жертвовать, как правило, во главу угла ставится производительность, ибо неторопливый процесс прорисовки карты обычно выводит пользователя из себя. Однако и простую карту можно оформить со вкусом. Кэшируемый сервис в этом плане более сбалансирован – сложность оформления влияет только на время построения кэша, но не его отображения.

Теперь рассмотрим ряд практических советов по выбору стилей отображения. Основным критерием будем считать читаемость условных обозначений. Вопросы быстродействия будут рассмотрены далее.

Контраст является универсальным и надежным способом обеспечить читаемость точечного, линейного или текстового символа на любом фоне. Если фон равномерный, то все просто. Достаточно подобрать контрастирующий с ним цвет. Если же фон непрерывно меняется (типичный пример – изображение рельефа отмывкой и послойной окраской), необходимо использовать в символе минимум 2 контрастирующих цвета. Если один начинает сливаться с фоном, второй обязательно будет читаться. Основные приемы – это тень, обводка (гало) и комплексный значок.

MapServer, как было продемонстрировано ранее, позволяет использовать антиалиасинг (сглаживание) при создании картографического изображения. Этот прием улучшает графическое качество изображения векторных объектов и текста, устраняя эффект ступенчатости наклонных линий. Однако он ощутимо увеличивает толщину тонких линий, а в случае использования прозрачности для объектов – изменяет цвет их границ. Читаемость мелкого текста также может сильно ухудшиться. Перед тем, как задействовать эту опцию, необходимо на небольшом фрагменте убедиться, что результат соответствует ожиданиям. И в случае необходимости подкорректировать размеры, толщины и цвета.

Формат растра. Для кэшируемых сервисов важен также тип используемого растра. Форматы со сжатием (например, JPEG) несколько “размыливают” изображение и привносят

разного рода артефакты. Здесь требуется оптимизация обозначений под формат растра – увеличение размеров и цветового контраста. Расчет кэша на небольшой фрагмент карты позволит проверить результаты.

Точечные объекты. Для значков на экране критичен размер. Он влияет и на читаемость самого значка, и на его отличимость от других. В тех случаях, когда форма является индикатором типа объекта (например, месторождения часто показываются черным цветом, но разными значками), минимальный допустимый размер значка – 6 пикселов. Если используется цветовое кодирование, и форма не столь важна, значки могут быть и меньше, но не менее 3 пикселов при условии хорошего контраста с фоном.

В MapServer можно конструировать комплексные значки из нескольких компонент. Типичный пример – значки населенных пунктов, которые обычно содержат светлый фоновый кружок и наложенный сверху темный значок из одной или нескольких концентрических окружностей.

Линейные и площадные объекты. Использование необоснованно жирных линий – это дурной тон. Толщина должна быть адекватна значимости объекта и, в то же время, быть минимально возможной при хорошей читаемости символа. Главным образом, это касается разного рода обводок (границы полигонов) и линейных объектов, не тяготеющих к площадным. Например, толщина проспектов может быть большой по сравнению с переулками, а вот повсеместно жирные речки смотрятся не очень.

При использовании антиалиасинга для кэшированного сервиса смысл приобретает дробное значение толщины в пикселах. При антиалиасинге линия дробится, и разница между толщиной в 0.5, 0.75 и 1 пиксел видна невооруженным глазом. Если в кэше желательно получить линию в 1 пиксел толщиной, необходимо использовать 0.5 пиксела в MapServer. Если использовать 1 пиксел, на выходе получится линия в 2-3 пиксела – весьма заметное и, как правило, нежелательное утолщение. Для линий толще 3 пикселов этот эффект не так ощутим.

Как и в случае с точечными объектами, использование контрастных символов должно обеспечить читаемость на любом фоне. Для дорог, например, этот прием весьма распространен. Однако, стоит помнить, что для таких символов расстояние между внешними элементами должно быть хотя бы 1.5 пиксела. Иначе они начнут сливаться с сердцевиной, особенно в случае антиалиасинга.

Шрифты и надписи. Для электронных карт наилучшим образом подходят шрифты без засечек (типа *Arial* или *Verdana*). Благодаря отсутствию мелких штриховых элементов на экране они воспринимаются легче, чем шрифты с засечками (типа Times New Roman или Georgia).

Допустимый минимальный размер шрифта зависит от многих факторов, среди которых начертание, насыщенность, контраст и т.д. Однако в целом для экрана можно рекомендовать кегль 8 и более пунктов. При использовании мелких шрифтов может потребоваться работа с кернингом. Незначительное увеличение интервала между буквами (на 5-10%) может существенно улучшить их читаемость, особенно при использовании антиалиасинга. Однако увлекаться кернингом не стоит, иначе он превращается в разрядку, а это уже способ выделения текста.

Используя для надписей обводку (гало) или тень, можно добиться их читаемости на любом фоне. Обводки используются, например, на картографических сервисах Google Maps и Яндекс.Карты. Здесь тоже надо знать меру – следует подбирать минимально возможный размер обводки, иначе буквы могут распухнуть. Для текста с кеглем порядка 8-14 пунктов вполне достаточно пары пикселов, чтобы обеспечить хороший контраст надписей с любым фоном. Если фон достаточно равномерный, то, скорее всего, можно обойтись и без обводок, подобрав соответствующий цвет для шрифта.

При подготовке карты по возможности следует использовать режим автоматизированной расстановки подписей. Это обеспечит их наилучшее размещение. После расстановки подписи желательно конвертировать в аннотации (статичные текстовые объекты).

Оптимизация

Оптимизация актуальна в первую очередь для динамических сервисов. Чем проще условные обозначения и структура атрибутивных полей слоев карты, тем быстрее она будет прорисовываться. В каждом слое необходимо отключить те атрибутивные поля, которые не используются для запросов (как пользовательских, так и участвующих в символогии). Оставшиеся поля необходимо проиндексировать в БД для ускорения запросов.

Структура проекта карты

Копируя и вставляя слои, группируя их и устанавливая масштабные диапазоны, можно добиться аккуратной и логичной структуры картографического проекта. Стандартное решение состоит в создании групп, соответствующих масштабам отображения, а внутри них – подгрупп, объединяющих объекты по тематике (рис. 4). Тематическая структура масштабных групп плавно изменяется. В этом случае диапазон масштабов отображения будет достаточно установить только для масштабных групп, а не для всех слоев.

Естественно, если запланировано 15 масштабов, вряд ли будет нормальным генерализовать каждый класс объектов 15 раз. Как правило, одной и той же степени геометрической подробности вполне достаточно для отображения на нескольких соседних масштабах. Поэтому слои в соседних масштабных группах могут ссылаться на один и тот же класс объектов. Однако может меняться симвология (допустим, с уменьшением масштаба дороги станут более тонкими), а также определяющий запрос (в примере с дорогами – отображать не все дороги, а только автомагистрали). Так что генерализация на самом деле есть, но происходит она уже при визуализации данных.

Заключение

Таким образом, создание качественной карты для web-сервиса – сложный и трудоемкий процесс, требующий внимания к разного рода мелочам и учета специфики среды отображения. Здесь необходима концентрация знаний по картографии, серверным технологиям, компьютерной графике, дизайну и цветоведению. Необходимо четкое понимание разработчиком задачи, которую требуется решить, тех ресурсов и источников данных, которые имеются, и тех технологий, которые позволяют с ними работать. В этом случае вопросы выбора типа сервиса, проекции, масштабного ряда, генерализации и оформления получат обоснованное решение.

2.9 Построение решения

Построение решения поставленной задачи будет сводиться к разработке архитектуры ИС. Архитектура ИС будет разрабатываться исходя из нескольких критериев – масштабируемости и производительности. Изначально, все компоненты ИС располагаются на одном сервере, исходя из этого, будет вестись дальнейшая разработка и оптимизация ИС. Но при этом необходимо учитывать возможность разнесения наиболее ресурсоемких компонентов ИС по разным серверам, для увеличения производительности и надежности. Поэтому, предложенный вариант архитектуры ИС должен быть пригоден для использования на одном сервере, а так же, с минимальными изменениями, для использования на нескольких серверах.

Исходя из структурного подхода к проектированию, постепенно будем разбивать существующие задачи на более мелкие компоненты. При таком нисходящем проектировании, в итоге, мы должны будем получить ту архитектуру, которая будет отвечать поставленным требованиям. А начнем с клиент-серверной природы разрабатываемого сервиса:

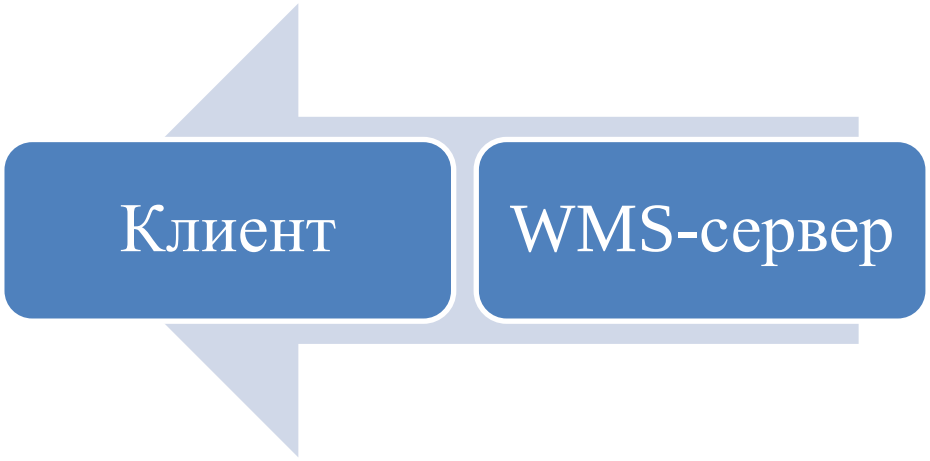


Рисунок 2.16 – Обобщенная архитектура web-сервиса

В простейшем случае, WMS-сервер передает клиенту результат своей работы. Детализируем компоненты клиента и wms-сервера:

Рисунок 2.17 – Детализированная архитектура web-сервиса

Пользователю необходимо получить информацию или изображение некоторых ГИС-объектов. Для этого, он должен выполнить, посредством wms-клиента, запрос к WMS-серверу, который, в свою очередь, вернет отрисованный результат из источника пространственных данных. Представленная структурная схема определяет три основных компонента, необходимых для визуализации данных, а что самое важное – эти *компоненты независимы друг от друга*.

Таким образом, следуя по пути более подробного рассмотрения схемы, получим следующую структурную схему:

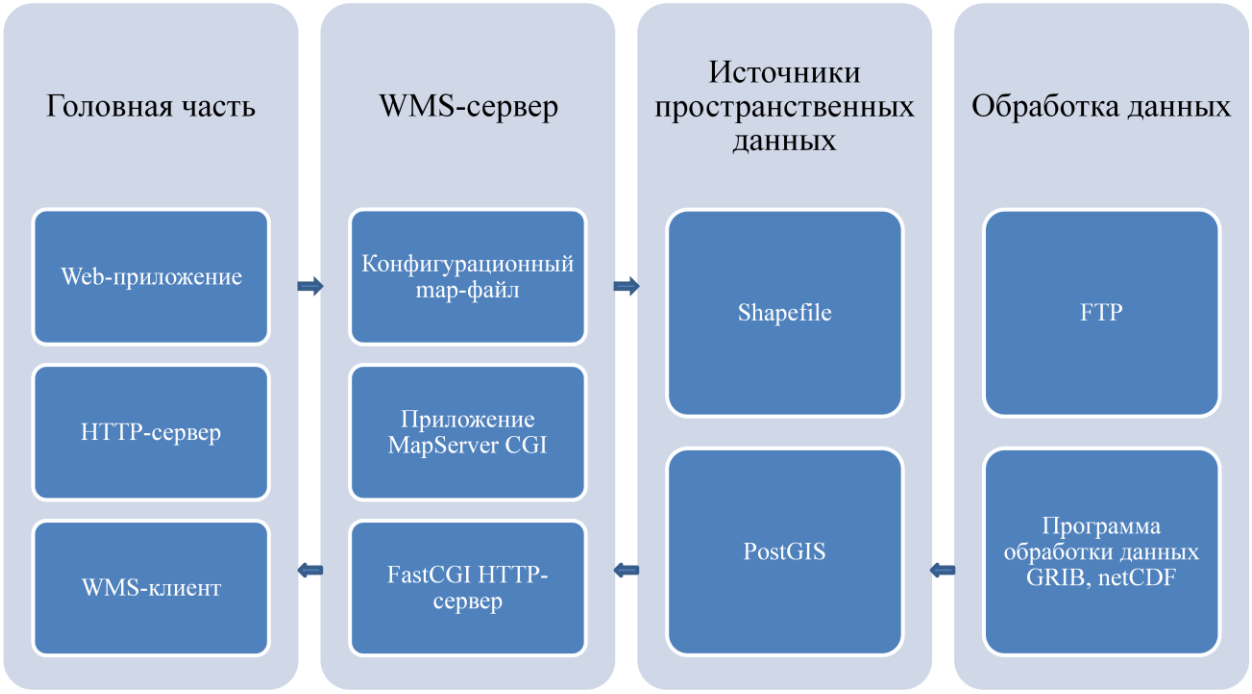


Рисунок 2.18 – Структурная схема web-сервиса

Как видно из рисунка 2.18, без компонента “Обработка данных”, структура представляет собой не что иное, как трехуровневую (трехзвенную) архитектуру. Такая архитектура предполагает наличие следующих компонентов приложения: клиентское приложение (“Головная часть”), сервер приложений (“WMS-сервер”) и база данных (“Источники пространственных данных”). Достоинства такой архитектуры, по сравнению с обычной клиент-серверной или файл-серверной, следующие:

- Масштабируемость;
- Конфигурируемость – изолированность уровней друг от друга позволяет (при правильном развертывании архитектуры) быстро и простыми средствами переконфигурировать систему при возникновении сбоев или при плановом обслуживании на одном из уровней;
- Высокая безопасность;
- Высокая надежность;
- Низкие требования к скорости канала;
- Низкие требования к производительности и техническим характеристикам клиентской части.

Конечно, есть и недостатки, как уже упоминалось, они связаны с более высокой сложностью создания приложений, их развертывания и администрирования. Но, достоинства от применения трехзвенной архитектуры, оправдывают эти сложности.

3 Описание практической части

Основные условия, компоненты и архитектура ИС определены, и можно переходить непосредственно к разработке сервиса. Как и прежде, задача разработки комплексной информационной системы визуализации и обработки геофизических данных, будет разбита на две подзадачи: визуализация данных и обработка данных. В разделе приводятся основные особенности настройки, полные исходные тексты программ и тексты конфигурационных файлов можно найти в “Приложение А”. Для выполнения дальнейших действий, определим базовую структуру каталогов проекта:

`/srv/wmsystem/` - домашний каталог пользователя `wmsystem`, от имени которого будет выполняться часть задач, решаемых ИС

`/srv/wmsystem/maps/` - каталог, содержащий необходимые конфигурационные файлы и данные для запуска MapServer

`/srv/wmsystem/web/` – каталог, содержащий web-приложение и его вспомогательные компоненты

`/srv/wmsystem/stream/` - каталог, содержащий программу обработки данных

Далее, будет рассмотрена более детальная структура каталогов, необходимые конфигурационные файлы, компоненты, и их местоположение. Следует сказать, что ниже рассматриваются наиболее значимые и важные части настроек, полные их версии слишком объемны и приводятся в приложении.

3.1 Задача обработки данных

Задача обработки данных сводится к разработке программы с использованием выбранных ранее инструментов, которая читает файлы во входной директории `/srv/wmsystem/stream/in/`. Запуск программы осуществляется по заранее определенному времени с помощью системного планировщика задач *Cron*. Файлы во входную директорию поступают с использованием протокола FTP (*File Transfer Protocol*), что соответственно, требует простейшей настройки и запуска FTP-сервера. Рассмотрим процесс запуска FTP-сервера:

```
/etc/init.d/vsftpd start
```

```
chkconfig vsftpd on
```

Теперь, обратившись с помощью FTP-клиента по адресу сервера, и указав логин и пароль ранее созданного пользователя, можно получить доступ к необходимому каталогу. *VsFTPd* по умолчанию предоставляет такой доступ для обычных пользователей системы, что очень удобно для проведения тестирования. Необходимости в более детальной настройке на данном этапе нет. Программа, выполняющая обработку данных, состоит из двух файлов: `in.py` – исполняемый файл; `grib.py` – библиотечный модуль.

3.1.1 Взаимодействие с PostGIS

Существует несколько подходов, которые позволяют сохранять данные GRIB в PostGIS:

- *Программный метод* – при использовании этого метода в программе задействуется интерфейс или драйвер для связи с базой данных, и вся подготовка и обработка запросов выполняется на уровне программы;
- *Дамп* – при работе с большими объемами данных, часто, в программе формируют *SQL-дамп*, а позже применяют его к базе данных.

Оба подхода могут обеспечить желаемый результат, но создание SQL-дампа является наиболее предпочтительным. Использование дампа, позволяет разделить нагрузку и ход выполнения программы, на два шага: подготовка дампа и вызов консольный утилиты для его обработки. Это не снижает гибкости и надежности, так как в дампе так же можно контролировать начало и конец транзакций и выполнять соответствующую обработку. Допустим, вставку значений каждого нового сообщения можно предварить командой *BEGIN* и закончить командой *COMMIT*.

Использование дампа для загрузки большого количества данных в БД, позволяет достичь результата быстрее, нежели использование интерфейсов и адаптеров из языков программирования. Это связано с более большими накладными расходами при установлении связи с БД, и контролем выполнения запросов.

Для создания пространственной привязки используется функция *ST_GeomFromText()* расширения PostGIS [8]. В качестве параметров выступают: геометрия объекта в формате WKT –

POINT(x, y), и значение географической проекции – 4326. Структура таблицы и необходимые настройки представлены ниже:

```
SET CLIENT_ENCODING TO UTF8;
SET STANDARD_CONFORMING_STRINGS TO ON;
SELECT DropGeometryColumn('','grib','the_geom');
DROP TABLE "grib";
BEGIN;
CREATE TABLE "grib" (gid serial PRIMARY KEY,
"initial_time" timestamp,
"center" varchar(254),
"parameter_number" int2,
"level_indicator" int2,
"level" int2,
"forecast_time" int2,
"grid_type" varchar(254),
"val" real);
SELECT AddGeometryColumn('','grib','the_geom','4326','POINT',2);
COMMIT;
```

SQL-командой, представленной ниже, можно создать пространственный индекс:

```
CREATE INDEX "poppnt_the_geom_gist" ON "grib" using gist ("the_geom"
gist_geometry_ops);
```

Пример записи, содержащей данные для какой-либо одной точки данных GRIB, может выглядеть следующим образом:

```
INSERT INTO "grib"
("center","val","level","level_indicator","forecast_time","initial_time","par
ameter_number","the_geom","grid_type") VALUES ('European Center for Medium-
Range Weather Forecasts (RSMC)',5851.215,500,100,168,'2011-01-27
(00:00)',7,ST_GeomFromText('POINT(0.0 0.0)', 4326),'Cylindrical Equidistant
Projection Grid');
```

3.1.2 Стратегии оптимизации

Программа будет располагаться и использоваться на том же сервере (в первоначальном варианте), что и остальные сервисы, поэтому, совершенно необходимо минимизировать

потребление ресурсов. Существует множество стратегий оптимизации [9] программ, написанных на Python, приведем основные из них:

- *Алгоритмы данных* – стратегия для разрабатываемой программы подразумевает использование математических пакетов и модулей, таких как NumPy, для обработки больших массивов данных.
- *Использование минимального уровня абстракций* – всякий раз, когда добавляется новый уровень абстракции или дополнительные удобства к функции или к объекту, это замедляет скорость работы программы. Необходимо сохранять баланс между удобством и производительностью.
- *Классы и экземпляры данных* – классы и экземпляры данных в Python основаны на применении словарей. По этой причине операции поиска, изменения или удаления данных в экземпляре практически всегда выполняются медленнее, чем непосредственные операции со словарями.
- *Использование специальных атрибутов* – такие атрибуты как *__slots__*, используемые в определении классов, ограничивают множество доступных имен атрибутов, и используют более эффективную внутреннюю структуру данных, что обеспечивает меньшее потребление памяти и эффективный доступ к данным.
- *Использование исключений* – использование исключений, это отличный способ не только добиться повышения производительности (блок try, в случае когда программный код не возбуждает исключение, выполняется быстрее чем if га 10%), но и обеспечить надежный перехват ошибок программы и предусмотреть их исправление.
- *Применение приемов функционального программирования* – генераторы списков, выражения-генераторы, функции-генераторы, сопрограммы и замыкания оказываются очень эффективными при работе с большими объемами данных, в некоторых случаях только использование генераторов, дает десятикратный прирост производительности, в сравнении с использованием для обхода элементов цикла *for*.

Данные стратегии позволят разработать эффективную программу обработки данных, с использованием языка программирования Python.

3.1.3 Библиотечный модуль

Библиотечный модуль `grib.py` содержит классы для извлечения и обработки сообщений GRIB:

- *Splitter* – класс извлекает сообщения GRIB из списка файлов и записывает их во временную директорию;
- *SplitterError* – класс-исключение для обработки ошибок извлечения сообщений;
- *Message* – класс предназначен для извлечения и первоначальной обработки дынных сообщения GRIB;
- *MessageError* – общий класс-исключение для перехвата любых ошибок, связанных с извлечением данных из сообщения GRIB;
- *MessageOpenError* – класс-исключение для обработки ошибок открытия сообщения GRIB;
- *MessageAttributeError* – класс-исключение для обработки ошибок, связанных с атрибутами данных сообщения GRIB;
- *MessageIncorrect* – класс-исключение для обработки ошибок, связанных с некорректными значениями в сообщении GRIB;
- *FilterMessage* – класс проверяет соответствие значений сообщения GRIB заданным параметрам;
- *FilterMessageFail* – класс исключение сигнализирует о несоответствии значения сообщения GRIB заданному параметру;
- *DumpToPostGIS* – класс обеспечивает преобразование данных GRIB и создание на их основе SQL-дампа для расширения PostGIS.

Каждый из классов библиотеки тестируется с использованием распространенной методики тестирования на основе модуля *unittest*.

3.1.4 Исполняемая программа

Исполняемая программа `in.py` содержит процедуры импорта необходимых классов из библиотечного модуля, и создания их объектов. Все операции с объектами и системные вызовы обернуты в инструкции *try*, что позволяет перехватывать не только определенные ранее исключения, но и ошибки времени выполнения. Данные меры позволяют простыми способами добиться необходимой устойчивости программы к сбоям.

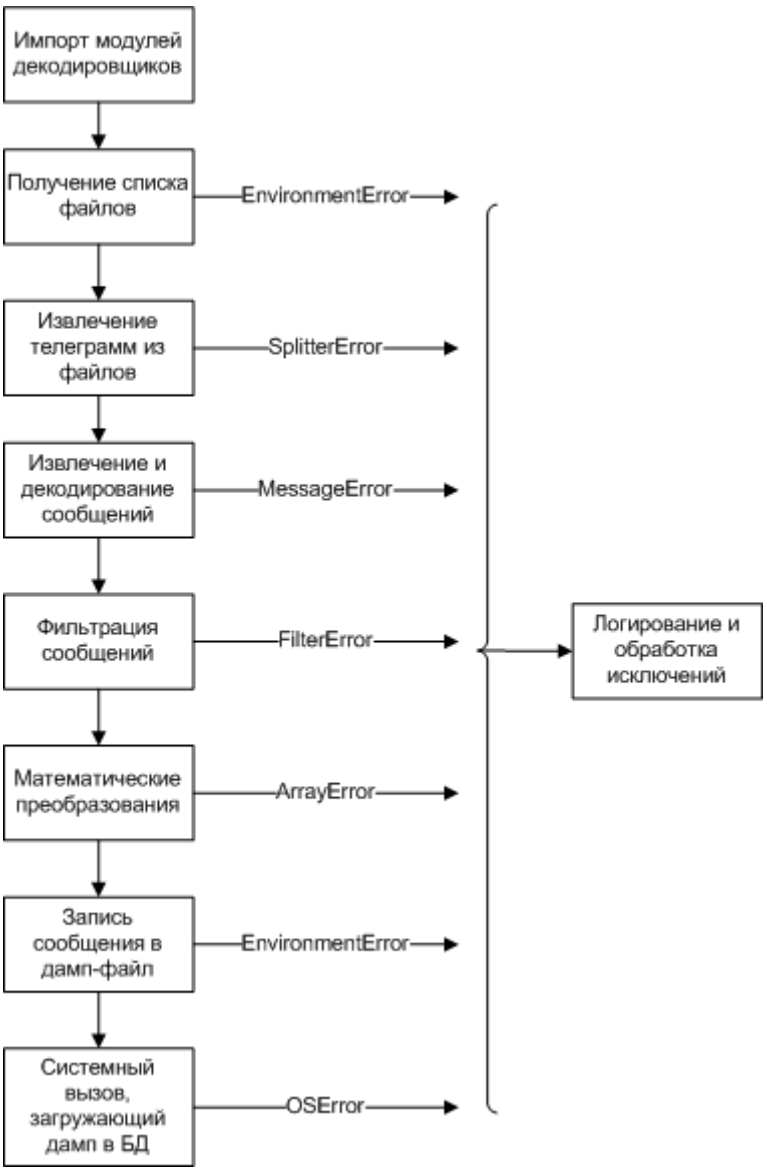


Рисунок 3.1 – Структура программы in.py

В программе и библиотечном модуле применены рассмотренные выше стратегии оптимизации, что позволило добиться хороших скоростных результатов. Время, затраченное на извлечение 12 сообщений GRIB, их декодирование, фильтрацию, создание и применение SQL-дампа, по данным UNIX утилиты *time*, представлено на рисунке 3.2.

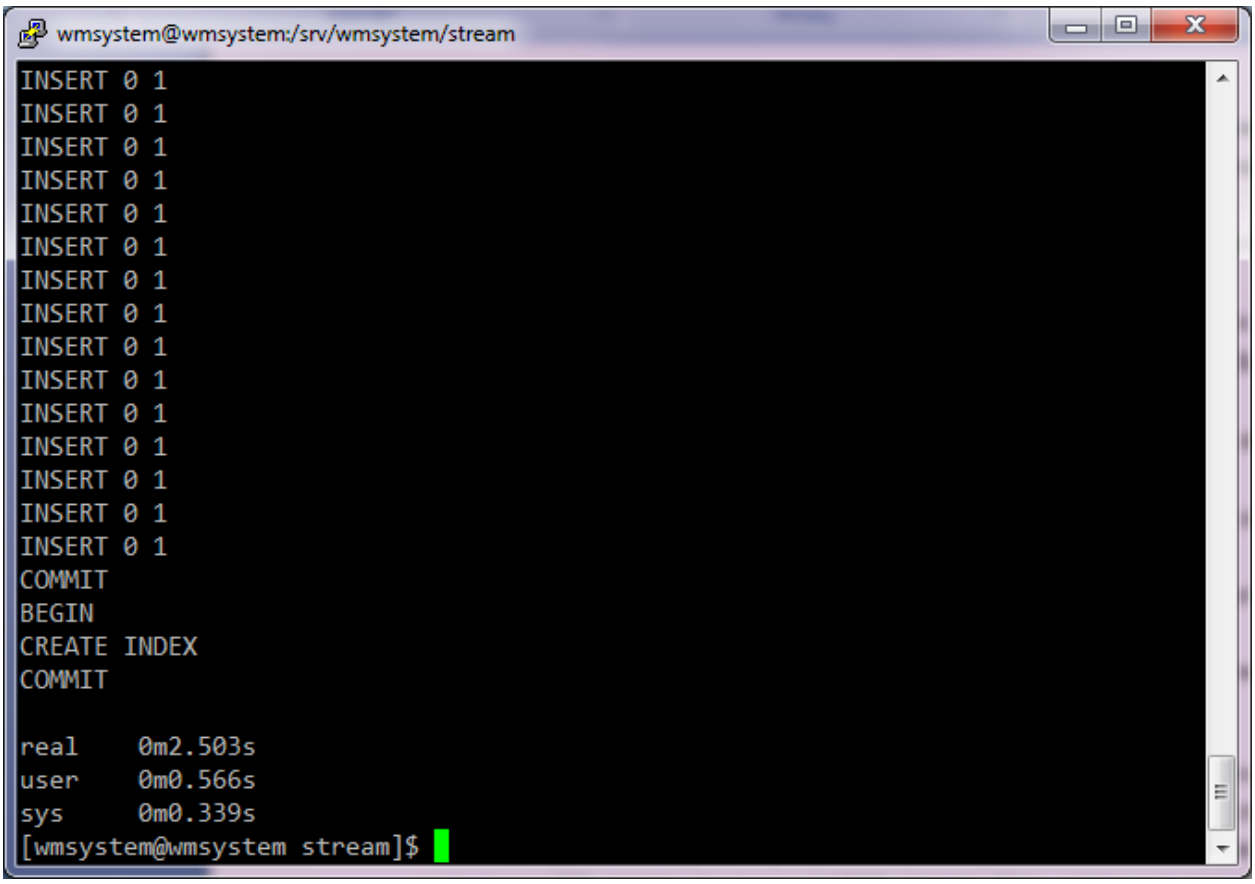


Рисунок 3.2 – Время выполнения программы обработки данных

Данные временные показатели удовлетворяют требуемой оперативности обработки данных. Так, программная обработка занимает всего лишь около 500 мс, остальное время приходится на заполнение таблицы в базе данных.

3.2 Задача визуализации данных

В данном разделе рассматриваются основные компоненты, необходимые для решения задачи визуализации данных, их взаимодействие и конфигурация. Задача будет решаться в два этапа:

- Выбор компонентов и их конфигурирование. Данный этап выполняется для одного слоя пространственных данных (сеточного).
- Подготовка слоев пространственных данных, и добавление их в уже сконфигурированную среду.

Такой подход обеспечит быструю разработку сервиса и получение результатов, нежели настройка web-сервиса одновременно для всех слоев пространственных данных. Выбор сеточного

слоя связан с тем, что он имеет минимальный объем, не требует дополнительной подготовки и позволяет наглядно оценивать результат выполняемых действий.

Фактически, после выполнения первого этапа, разработка системы визуализации данных будет закончена, дальнейшие действия будут сводиться к добавлению названий слоев подготовленных пространственных данных в конфигурационные файлы, и настройке их представления, что не представляет особой сложности.

3.2.1 Конфигурация WMS-сервера

Простейший вариант конфигурации wms-сервера, на базе MapServer, был описан в проведенном исследовании. Для дальнейших манипуляций, следует определиться со структурой каталогов wms-сервера:

`/srv/wmsystem/maps` – базовый каталог для хранения данных и конфигурации карты

`/srv/wmsystem/maps/data` – каталог для пространственных данных в формате Shapefile

`/srv/wmsystem/maps/templates` – каталог для хранения шаблонов для обработки запросов *GetFeatureInfo*, которые служат для получения атрибутивной информации

`/srv/wmsystem/maps/fonts` – каталог для хранения шрифтов подписей объектов ГИС

`/srv/wmsystem/maps/wms.map` – map-файл, содержащий настройки карты и слоев

`/tmp/mapserv/tmp` – каталог для хранения временных файлов

`/tmp/mapserv/debugs.log` – лог-файл MapServer, содержит такие полезные данные, как время генерации слоя и возможные ошибки, что очень полезно при отладке сервиса

Для получения атрибутивной информации в протоколе WMS описан такой запрос как *GetFeatureInfo*. В MapServer результат ответа на этот запрос может быть представлен в одном из трех форматов:

- *text/plain* - простой текст (используется по умолчанию)
- *text/html* - текст в формате html
- *application/vnd.ogc.gml* - ответ в формате GML

Каждый из этих форматов требует отдельной настройки, и имеет свои недостатки и достоинства. Полное описание этих форматов можно найти в документации к MapServer. Наиболее удобен, с точки зрения представления информации, формат *text/html*. Он позволяет определить три типа шаблонов (*HEADER, FOOTER, TEMPLATE*), которые будут отвечать за представление атрибутивной информации. Файлы шаблонов HEADER и FOOTER обрабатываются единожды, в то время как обращение к файлу TEMPLATE происходит каждый раз по мере извлечения записей из источника данных.

Ниже приведен адаптированный map-файл, который способен не только отображать картографическую информацию, но и обеспечивать получение атрибутов ГИС-объектов:

```
MAP


NAME        "WMSYSTEM_WMS"
STATUS      ON
SIZE        800 600
EXTENT      -5133549.567108 -5397733.446068 5397733.446068 1667991.365941
UNITS       METERS
SHAPEPATH   "data/"
IMAGECOLOR  255 255 255
CONFIG      "PROJ_LIB" "/usr/share/proj"
CONFIG      "MS_ERRORFILE" "/tmp/mapserv/debugs.log"
DEBUG       5



#
# Web-определения
#

WEB
    HEADER "templates/header.html"
    TEMPLATE "templates/template.html"
    FOOTER "templates/footer.html"
    IMAGEPATH "/tmp/mapserv/tmp/"
```

```
    IMAGEURL "/tmp/"
    METADATA
        "mapfile_encoding" "UTF-8"
        "wms_title" "WMSYSTEM_WMS"
        "wms_onlineresource"
"http://localhost:8088/maps?map=/srv/wmsystem/maps/wms.map&"
        "wms_srs" "EPSG:3576 EPSG:4326"
        "wms_getfeatureinfo"
"http://localhost:8088/maps?map=/srv/wmsystem/maps/wms.map&"
        "wms_featureinfoformat" "text/html"
        "wms_feature_info_mime_type" "text/html"
    END
END

#
# Проекция карты
#

PROJECTION
    "init=epsg:3576"
END

#
# Определение формата PNG со сглаживанием
#

OUTPUTFORMAT
    NAME aggpng24
    DRIVER AGG/PNG
    MIMETYPE "image/aggpng24"
    IMAGEMODE RGB
    EXTENSION "png"
END
```

```
#
# Определения слоев
#

LAYER
    NAME "crdlin"
    METADATA
        "wms_title" "Сетка"
        "wms_abstract" "Grid 1 test"
        "wms_srs" "EPSG:3576 EPSG:4326"
        "wms_include_items" "all"
    END
    TYPE LINE
    HEADER "templates/header.html"
    TEMPLATE "templates/template.html"
    FOOTER "templates/footer.html"
    STATUS ON
    DATA crdlin
    PROJECTION
        "init=epsg:3576"
    END
    CLASS
        NAME "Сетка"
        STYLE
            COLOR 255 0 0
        END
    END
END

LAYER
    NAME "grib"
```

```
    METADATA
        "wms_title" "Сетка"
        "wms_abstract" "Grid 1 test"
        "wms_srs" "EPSG:3576 EPSG:4326"
        "wms_include_items" "all"
    END
    CONNECTIONTYPE postgis
    CONNECTION "user=postgres dbname=wmsystem host=127.0.0.1"
    DATA "the_geom from grib"
    TYPE POINT
    HEADER "templates/header.html"
    TEMPLATE "templates/template.html"
    FOOTER "templates/footer.html"
    STATUS ON
    PROJECTION
        "init=epsg:4326"
    END
    CLASS
        NAME "Сетка"
        STYLE
            COLOR 0 0 0
        END
    END
END


END
```

Настройка остальных слоев аналогична, и сводиться больше к дизайнерской работе, нежели разработке, поэтому в данном разделе не описывается, а приводится полностью в приложении.

3.2.2 Конфигурация WMS-клиента

На сегодняшний день существует лишь одна библиотека, удовлетворяющая условиям, выдвинутым ранее, и позволяющая создавать полноценные web-приложения - *OpenLayers*. OpenLayers – библиотека с открытым исходным кодом [10], написанная на JavaScript, предназначенная для создания карт на основе программного интерфейса (API), подобного GoogleMap API или MSN Virtual Earth API. Библиотека позволяет очень быстро создать web-интерфейс для отображения картографических материалов, представленных в различных форматах и расположенных на различных серверах. Благодаря OpenLayers разработчик имеет возможность создать, к примеру, собственную карту, включающую слои, предоставляемые WMS (и WFS) серверами, такими как MapServer, ArcIMS или GeoServer, и данными картографических серверов Google. На базе этой библиотеки, разрабатывается множество других продуктов, как правило, нацеленных на решение узкого круга задач.

OpenLayers является набором скриптов, написанных на JavaScript. Эта библиотека не требует установки в привычном смысле слова. Чтобы начать работать с OpenLayers, достаточно загрузить библиотеку с официального сайта и распаковать ее в необходимый каталог. С библиотекой поставляется множество дополнительных материалов: документация, примеры использования, средства оптимизации, и т.п. Для нормальной работы достаточно скопировать в каталог, в котором хранится проект, файл *build/OpenLayers.js* и каталоги *theme* и *img*, на этом установка будет завершена.

В OpenLayers API есть два основополагающих понятия, идентичных аналогичным предложениям Mapserver: это объекты *Map* и *Layer*. Map хранит информацию о проекции, географическому охвату, единицах измерения и других параметрах карты в целом. Внутри карты (Map) данные задаются посредством одного или нескольких объектов Layer. Layer содержит информацию о слоях данных, которые будут помещены на карту, и о том, как каждый из этих слоев должен отображаться на карте.

Для начала нужно подготовить исходный файл, в который будет встроен объект OpenLayers, отвечающий за отображение карт (OpenLayers поддерживает встраивание карт в любой блочный элемент html-кода). Кроме этого, в текст web-страницы нужно вставить тег со ссылкой на скрипт из библиотеки OpenLayers:

```
<html>
<head>
  <title>OpenLayers Example</title>
```

```
    <script
    src="http://openlayers.org/api/OpenLayers.js"></script>
    </head>
    <body>
      <div style="width:100%; height:100%" id="map"></div>
      <script defer="defer" type="text/javascript">
        var map = new OpenLayers.Map('map');
        var wms = new OpenLayers.Layer.WMS( "OpenLayers WMS",
            "http://labs.metacarta.com/wms/vmap0", {layers: 'basic'} );
        map.addLayer(wms);
        map.zoomToMaxExtent();
      </script>
</body>
</html>
```

Данного примера достаточно, чтобы отобразить один слой открытого картографического сервера MetaCarta. Библиотека имеет очень большое количество настроек и классов для работы с картографическими объектами, описание которых можно найти в документации.

Для решения поставленной задачи визуализации данных, потребуется две конфигурации OpenLayers:

- *Для основной проекции EPSG:3576*:

```
var options = {
    controls: [
        new OpenLayers.Control.Navigation(),
        new OpenLayers.Control.PanZoomBar(),
        new OpenLayers.Control.LayerSwitcher({'ascending':false}),
        new OpenLayers.Control.Permalink(),
        new OpenLayers.Control.ScaleLine(),
        new OpenLayers.Control.Permalink('permalink'),
        new OpenLayers.Control.MousePosition(),
        new OpenLayers.Control.OverviewMap(),
        new OpenLayers.Control.KeyboardDefaults(),
```

```
        new OpenLayers.Control.NavToolbar()
    ],
    scales: [50000000, 35000000, 25000000, 10000000, 3000000, 1000000,
500000, 300000, 200000, 100000, 50000, 25000, 10000, 5000, 2000],
    maxExtent: new OpenLayers.Bounds(-5133549.56760757, -5397733.44656832,
5397733.44656832, 1667991.36644116),
    maxResolution: "auto",
    projection: "EPSG:3576",
    units: "m"
};


var map = new OpenLayers.Map("map", options);
var wms = new OpenLayers.Layer.WMS("Основа", server, {layers: "crdlin",
format: "aggpng24"}, {singleTile: true} );


map.addLayer(wms);


info = new OpenLayers.Control.WMSGetFeatureInfo({
    url: server,
    title: 'Identify features by clicking',
    queryVisible: true,
    eventListeners: {
        getfeatureinfo: function(event) {
            map.addPopup(new OpenLayers.Popup.FramedCloud(
                "chicken",
                map.getLonLatFromPixel(event.xy),
                null,
                event.text,
                null,
                true
            ));
        }
    }
});
```

```
map.addControl(info);
info.activate();

if (!map.getCenter()) map.zoomToMaxExtent();
```

- *Для дополнительной проекции EPSG:4326*:

```
var options = {
    controls: [
        new OpenLayers.Control.Navigation(),
        new OpenLayers.Control.PanZoomBar(),
        new OpenLayers.Control.LayerSwitcher({'ascending':false}),
        new OpenLayers.Control.Permalink(),
        new OpenLayers.Control.ScaleLine(),
        new OpenLayers.Control.Permalink('permalink'),
        new OpenLayers.Control.MousePosition(),
        new OpenLayers.Control.OverviewMap(),
        new OpenLayers.Control.KeyboardDefaults(),
        new OpenLayers.Control.NavToolbar()
    ],
    scales: [100000000, 50000000, 35000000, 25000000, 10000000, 3000000,
1000000, 500000, 300000, 200000, 100000, 50000, 25000, 10000, 5000, 2000],
    maxExtent: new OpenLayers.Bounds(-180.00, 40.00, 180.00, 84.00),
    maxResolution: "auto",
    projection: "EPSG:4326",
    units: "dd"
};

var map = new OpenLayers.Map("map", options);
var wms = new OpenLayers.Layer.WMS("Основа", server, {layers: "crdlin",
format: "aggpng24"}, {singleTile: true} );

map.addLayer(wms);
```

```
info = new OpenLayers.Control.WMSGetFeatureInfo({
    url: server,
    title: 'Identify features by clicking',
    queryVisible: true,
    eventListeners: {
        getfeatureinfo: function(event) {
            map.addPopup(new OpenLayers.Popup.FramedCloud(
                "chicken",
                map.getLonLatFromPixel(event.xy),
                null,
                event.text,
                null,
                true
            ));
        }
    }
});
map.addControl(info);
info.activate();

if (!map.getCenter()) map.zoomToMaxExtent();
```

Обе конфигурации выполняют запрос к wms-серверу, который задается в переменной *server*. Добавить новые слои для отображения можно простым перечислением в ключе хэша *layers*, формат картографического результата задается ключом *format.* Этого простого определения достаточно, что бы отобразить необходимые слои данных. Необходимо отметить присутствие объекта *OpenLayers.Control.WMSGetFeatureInfo*, который отвечает за получение атрибутивной информации в формате text/html, который был рассмотрен при настройке wms-сервера. Полная версия данных конфигураций содержится в приложении.

3.2.3 Разработка web-приложения

В качестве основы для разрабатываемого web-приложения будет использоваться написанная на Python библиотека *Web.py* [11]. Выбор обусловлен характером разрабатываемого сервиса, в нем не планируется использование новостной системы или форума, его главная задача – загрузка wms-клиента пользователю с необходимыми параметрами. Web.py позволяет строить легкие web-приложения, с минимальными временными затратами и потребляемыми ресурсами. При этом в нем есть все необходимые средства, будь то шаблоны, механизмы обработки данных пользователя, сессии, авторизация и другие.

Для установки web.py, необходимо с помощью утилиты easy_install, установить соответствующий пакет:

```
easy_install web.py
```

Структура каталогов проекта web-приложения выглядит следующим образом:

`/srv/wmsystem/www` – базовый каталог web-приложения

`/srv/wmsystem/www/static` – каталог для статического содержимого, такого как CSS-стили и JavaScript-скрипты

`/srv/wmsystem/www/templates` – шаблоны web-приложения

`/srv/wmsystem/www/app.py` – исполняемый файл

`/srv/wmsystem/www/start.sh` – скрипт для запуска web-приложения

`/srv/wmsystem/www/stop.sh` – скрипт для остановки web-приложения

На официальном сайте web.py предоставлена исчерпывающая информация о возможностях библиотеки. Ниже приводятся простейшая конфигурация, которой достаточно для обеспечения визуализации данных:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
#

import os
import sys
import web
```

```
render = web.template.render('templates/', base='layout')
urls = (
    "/", "IndexHandler",
    "/map/(.*)", "MapHandler",
)
app = web.application(urls, globals())

class IndexHandler:
    def GET(self):
        raise web.seeother('/map/')

class MapHandler:
    def GET(self, proj):
        i = web.input(proj="epsg:3576")
        return render.map(i.proj)

if __name__ == '__main__':
    web.wsgi.runwsgi = lambda func, addr=None: web.wsgi.runfcgi(func, addr)
    app.run()
```

Данное web-приложение обеспечивает доступ к карте по относительному адресу /map/, и подключение, с помощью механизма шаблонов, соответствующей конфигурации OpenLayers для отображения карты в требуемой проекции. Исходный код шаблонов приводится в приложении.

Запуск web-приложения осуществляется в с помощью утилиты *spawn-cgi*. Spawn-CGI является частью проекта Lighttpd, и предназначен для запуска приложения как сервера FastCGI, с которым может работать потом практически любой HTTP-сервер. Для запуска web-приложения используется скрипт `start.sh`:

```
#!/bin/sh
spawn-fcgi -d /srv/wmsystem/www/ -f /srv/wmsystem/www/app.py -a 127.0.0.1 -p 9002
```

Spawn-CGI запускает процесс, который прослушивает порт 9002, завершить этот процесс можно с помощью скрипта `stop.sh`:

```
#!/bin/sh
kill `pgrep -f "python /srv/wmsystem/www/app.py"`
```

Таким образом, web-приложение работает в собственном окружении, в виде обособленного компонента.

3.2.4 Настройка web-сервера

В разрабатываемой информационной системе уже есть несколько HTTP-серверов, которые обеспечивают взаимодействие с теми или иными компонентами системы. В качестве головного web-сервера, должен выступать легкий сервер, ориентированный на обработку большого количества запросов при минимальном использовании системных ресурсов. Это позволит, при возможном масштабировании системы, не менять его, а внести всего лишь незначительные изменения в конфигурацию. Так же, он должен обеспечивать возможности проксирования запросов, балансировки нагрузки и обеспечивать достаточный уровень надежности.

Можно выделить два популярных сервера, удовлетворяющих этим требованиям: *Lighttpd* и *Nginx*. Сервер Lighttpd уже используется для обеспечения взаимодействия с MapServer. При этом можно, конечно, внести необходимые изменения в его конфигурацию и заставить работать в качестве головного web-сервера, но это будет не правильно с точки зрения архитектуры сервиса, так как потребует значительного переконфигурирования при масштабировании системы.

Nginx – web-сервер и почтовый прокси-сервер, разрабатываемый для компании Рамблер [12]. Используется на большом количестве самых посещаемых сайтов в мире. В Nginx рабочие процессы обслуживают одновременно множество соединений, мультиплексируя их вызовами операционной системы *select, epoll* (Linux) и *kqueue* (FreeBSD). Рабочие процессы выполняют цикл обработки от дескрипторов. Полученные от клиента данные разбираются с помощью конечного автомата. Разобранный запрос последовательно обрабатывается цепочкой модулей, задаваемых конфигурацией. Ответ клиенту формируется в буферах, которые хранят данные либо в памяти, либо указывают на отрезок файла. Буферы объединяются в цепочки, определяющие последовательность, в которой данные будут переданы клиенту. Если операционная система поддерживает эффективные операции ввода-вывода, такие как *writev* и *sendfile*, то nginx применяет их по возможности.

По данным Netcraft за февраль 2011 года, число сайтов, обслуживаемых Nginx, превышает 21,57 миллионов, что делает его третьим по популярности web-сервером в мире. Как и Lighttpd,

Nginx часто используют для отдачи статического содержимого, генерируемого тяжелым web-приложением, работающим под управлением другого web-сервера. Среди известных проектов, использующих Nginx: Rambler, Yandex, Begun, Wordpress.com, SourceForge.net, vkontakte.ru, и многие другие.

Данные качества делают Nginx хорошим выбором, для решения поставленных задач. При этом сохраняется возможность быстрого масштабирования информационной системы. Настройка Nginx будет заключаться в настройке взаимодействия между тремя компонентами: web-приложением, wms-сервером, хранилищем статических файлов. Ниже приводится фрагмент конфигурационного файла `/etc/nginx/nginx.conf`, ответственный за запуск подготовленного web-приложения:

```
location / {
        fastcgi_param REQUEST_METHOD $request_method;
        fastcgi_param QUERY_STRING $query_string;
        fastcgi_param CONTENT_TYPE $content_type;
        fastcgi_param CONTENT_LENGTH $content_length;
        fastcgi_param GATEWAY_INTERFACE CGI/1.1;
        fastcgi_param SERVER_SOFTWARE nginx/$nginx_version;
        fastcgi_param REMOTE_ADDR $remote_addr;
        fastcgi_param REMOTE_PORT $remote_port;
        fastcgi_param SERVER_ADDR $server_addr;
        fastcgi_param SERVER_PORT $server_port;
        fastcgi_param SERVER_NAME $server_name;
        fastcgi_param SERVER_PROTOCOL $server_protocol;
        fastcgi_param SCRIPT_FILENAME $fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_script_name;
        fastcgi_pass 127.0.0.1:9002;
}
```

Организация отдачи статических файлов выполняется следующим образом:

```
        location /static/ {
                root /srv/wmsystem/www;
                if (-f $request_filename) {
                        rewrite ^/static/(.*) /static/$1 break;
```

```
        }
    }
```

При обращении по относительному адресу */static/*, будет доступно содержимое каталога `/srv/wmsystem/www/static/`.

Необходимо более подробно описать взаимодействие web-приложения и wms-сервера, посредством Nginx. Библиотека OpenLayers при осуществлении запроса типа GetFeatureInfo запрещает передавать такие запросы wms-серверу, если они осуществляются с другого ip-адреса или доменного имени. Такое ограничение служит защитой от атаки типа XSS (*Cross Site Scripting* – межсайтовый скриптинг). Для того, что бы запросы этого типа работали, необходимо “замаскировать” wms-сервис. Это также избавит нас от дополнительной перенастройки при смене ip-адреса сервера, и будет хорошим решением с точки зрения архитектуры:

```
location /maps {
            proxy_set_header X-Forwarded-Host $host;
            proxy_set_header X-Forwarded-Server $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_pass http://localhost:8088/maps;
    }
```

Таким образом, по относительному адресу */maps* будет доступен wms-сервис информационной системы, и пользователь, посредством web-приложения, сможет получить доступ к нему. На этом базовая конфигурация Nginx закончена, с содержанием конфигурационного файла которого, можно ознакомиться в приложении. В правильности выполненных настроек можно убедиться, выполнив в браузере переход по адресу сервера.

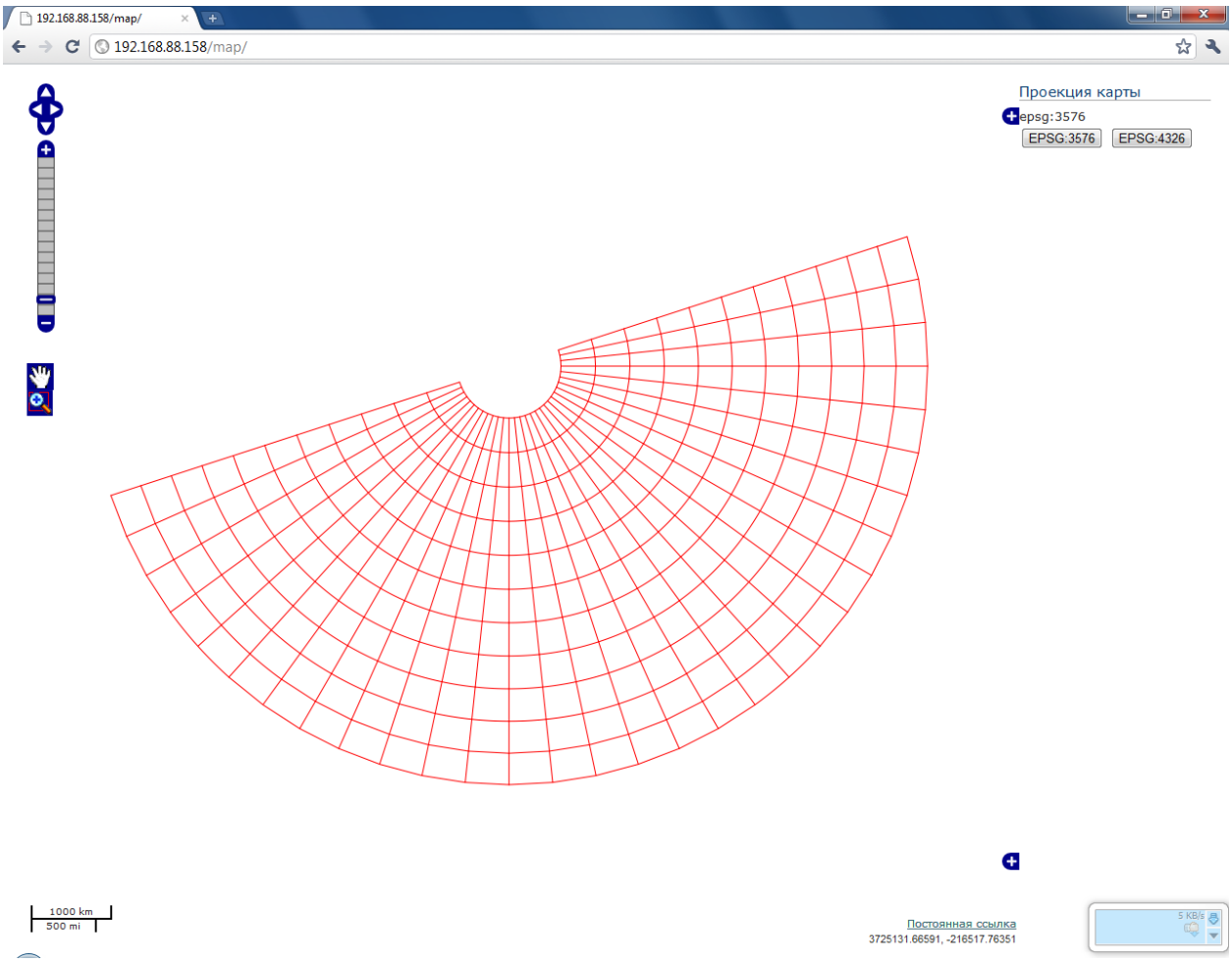


Рисунок 3.3 – Результат настройки web-приложения для проекции EPSG:3576

На рисунке 3.3 представлен результат настройки web-приложения и необходимых компонентов, на примере сеточного слоя. Для отображения карты в проекции EPSG:4326, необходимо нажать на соответствующую кнопку в панели справа. Результат представлен на рисунке 3.4.

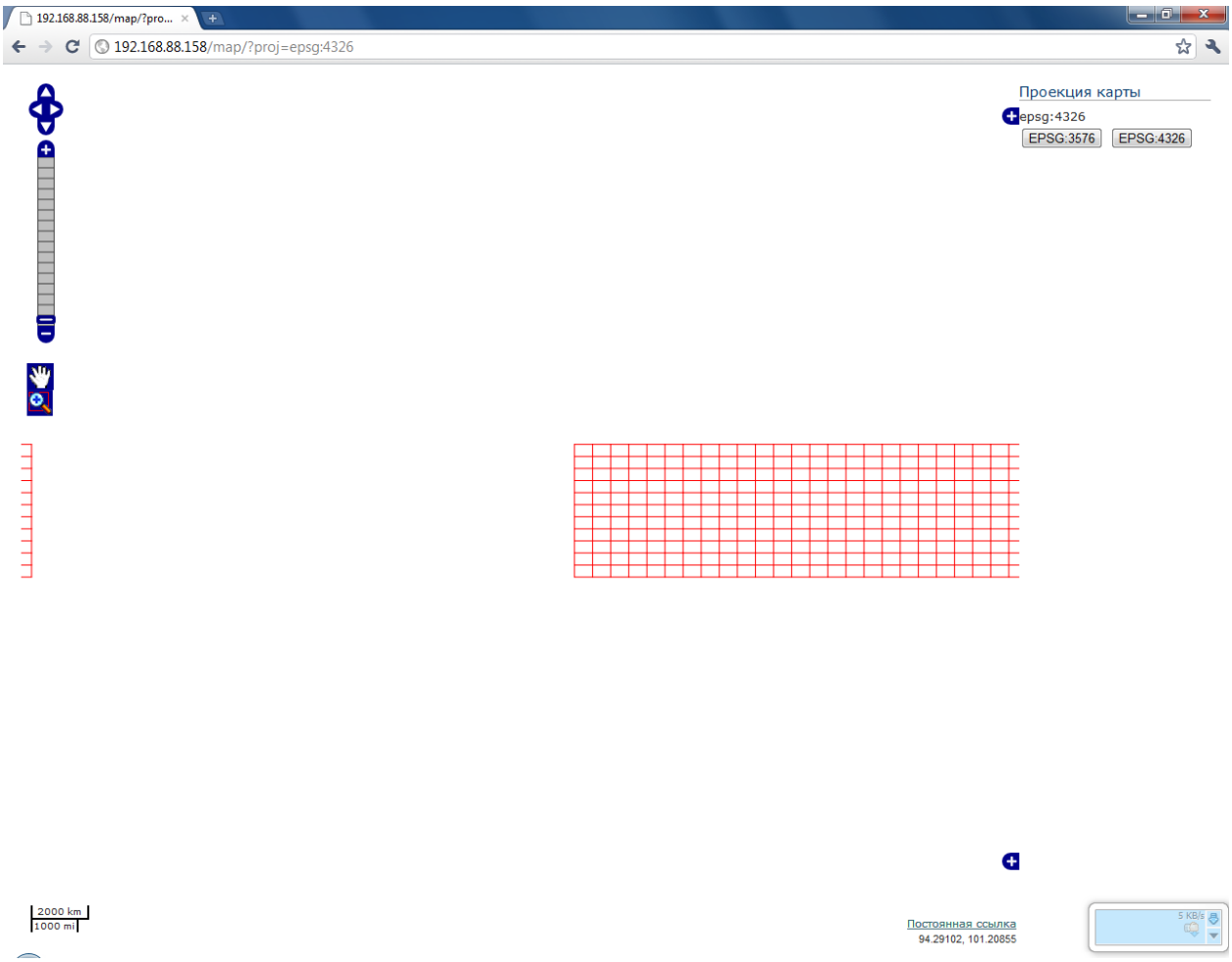


Рисунок 3.4 – Результат настройки web-приложения для проекции EPSG:4326

Для проверки работоспособности слоя PostGIS, следует подключить его в конфигурации OpenLayers, добавив в список слоев значение *grib*. Результат выполненного действия, представлен на рисунке 3.5.

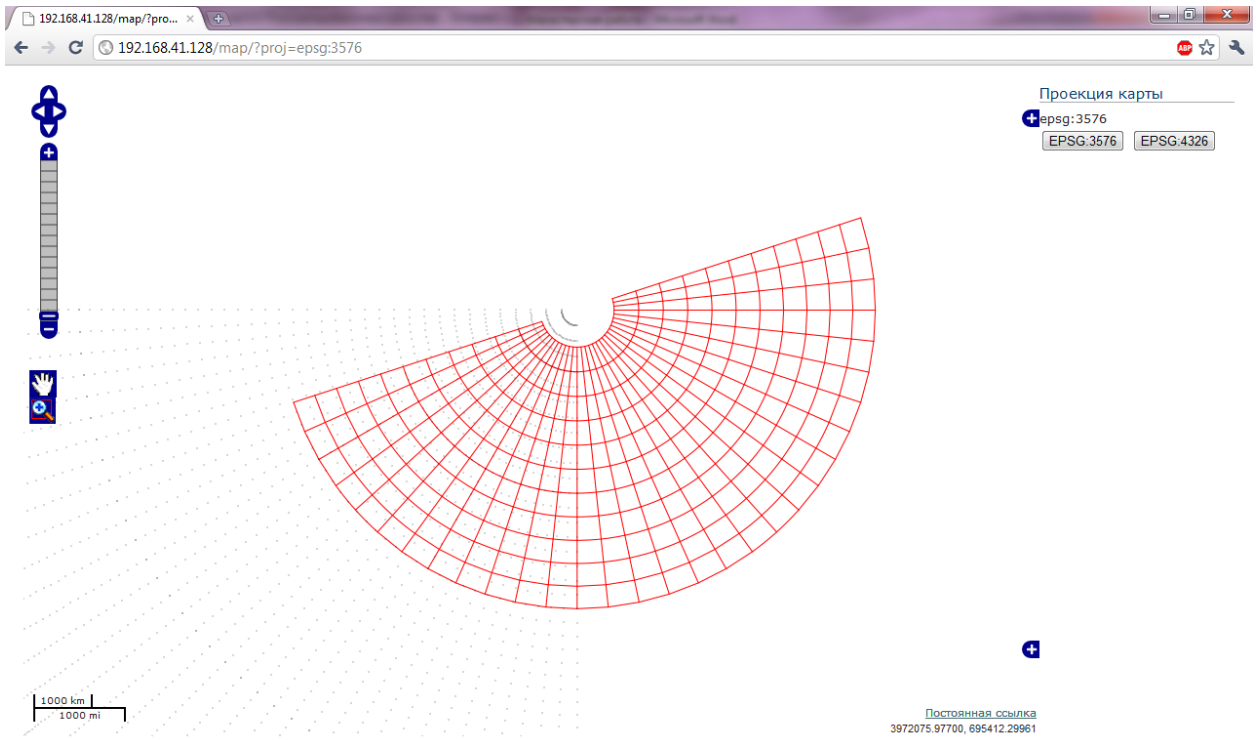


Рисунок 3.5 – Результат настройки web-приложения для проекции EPSG:3576, с подключенным слоем GRIB

Несмотря на то, что ни для одного из представленных слоев пока не настраивались стили отображения, видно, что сетка данных присутствует для половины части северного полушария.

Итак, мы убедились в правильности настройки компонентов визуализации данных на примере одного слоя. Далее будут рассмотрены этапы подготовки остальных данных, и представлен конечный результат.

3.2.5 Подготовка пространственных данных

Ранее упоминалось, что пространственные данные перед их непосредственным использованием необходимо предварительно подготовить. Это относится к пространственным данным в формате Shapefile, которые используются для создания основы карты. Эти данные содержат следующие слои:

- Сеточный слой (линейный тип);
- Административные границы (линейный тип);
- Административные границы (полигональный тип);
- Гидрография (линейный тип);
- Гидрография (полигональный тип);

- Населенные пункты (точечный тип).

Подготовка этих слоев данных будет сводиться к следующим операциям:

- Генерализация – разбивка слоя пространственных данных на несколько слоев согласно масштабной линейке, с целью уменьшения нагрузки на картографический сервер;
- Создание индексов для пространственных данных слоя;
- Оптимизация атрибутивных полей в базе данных слоя.

Генерализация выполняется в одной из настольных ГИС, например Quantum GIS, и заключается в разбиении исходного слоя на несколько более мелких слоев. Например, города ранжируются по численности населения, поэтому слой с населенными пунктами разумно разбить на несколько слоев по рангу, допустим города с населением больше 100000, больше 10000, и т.д. Это избавит картографический сервер от операции сортировки и выборки данных из исходного слоя при переходе на другой масштаб. Данная операция носит экспериментальный характер, и требует знаний картографии, поэтому рассматривать ее ход в данной работе не имеет смысла.

Создать индексы для пространственных данных можно также в настольной ГИС. Ниже приводятся команды для создания индексов с помощью уже рассмотренной выше библиотеки GDAL/OGR:

`shptree layer.shp,` - в простейшем случае создаст файл с пространственными индексами для слоя `layer.shp`

`sortshp layer.shp layer-sort.shp class1 ascending,` – создаст отсортированный слой с пространственными данными по полю `class1`

Таким образом, с помощью перечисленных действий пространственные данные подготавливаются для публикации в web, и можно рассчитывать на более быстрое их отображение.

3.3 Оптимизация web-приложения

Созданное ранее web-приложение содержит довольно большой объем загружаемых пользователю библиотек, так, OpenLayers в исходном варианте занимает около 1Мб. В данном разделе приводятся рекомендации, с помощью которых можно ускорить загрузку и инициализацию web-карты:

- Перестроение и оптимизация библиотеки OpenLayers;
- Сжатие статического содержимого.

В комплекте с библиотекой OpenLayers поставляется скрипт `build.py`, который выполняет подключение только необходимых модулей и возможностей, а так же сжатие результирующего js-кода. Составим конфигурацию для build.py, и сохраним ее в файле `wmsystem.cfg`:

```
[first]
OpenLayers/SingleFile.js
OpenLayers.js
OpenLayers/BaseTypes.js
OpenLayers/BaseTypes/Class.js
OpenLayers/Util.js

[last]

[include]
OpenLayers/Map.js
OpenLayers/Layer/WMS.js

OpenLayers/Control/Navigation.js
OpenLayers/Control/PanZoomBar.js
OpenLayers/Control/Permalink.js
OpenLayers/Control/ScaleLine.js
OpenLayers/Control/MousePosition.js
OpenLayers/Control/OverviewMap.js
OpenLayers/Control/KeyboardDefaults.js

OpenLayers/Control/WMSGetFeatureInfo.js
OpenLayers/Format/WMSGetFeatureInfo.js
OpenLayers/Request/XMLHttpRequest.js
OpenLayers/Format/GML.js
OpenLayers/Popup/FramedCloud.js
```

```
[exclude]
```

После выполнения команды `./build.py wmsystem.cfg`, получим оптимизированный файл библиотеки OpenLayers, сжатый с помощью инструмента *jsmin*. Таким образом, размер библиотеки OpenLayers уменьшился с 1Мб до 290Кб, что существенно ускоряет время ее загрузки пользователю.

Вторая мера, связанная со сжатием статических данных на лету, так же позволяет уменьшить количество трафика передаваемого пользователю. Однако необходимо отметить, что данная мера повышает нагрузку на CPU сервера, поэтому ее можно рекомендовать только для многоядерных конфигураций. Эта мера связана с подключением и базовой настройкой модуля gzip для nginx [13]:

```
gzip_static on;
gzip on;
gzip_comp_level 9;
gzip_types application/x-javascript text/css image/png;
```

Такая конфигурация заставит nginx сжимать не только статическое содержимое, в виде css и js-файлов, но и сгенерированные wms-сервером участки картографического изображения. Рекомендуется задействовать несколько процессов nginx, для обеспечения более высокой производительности в работе с данным модулем:

```
worker_processes  2;
```

С помощью данных мер, можно уменьшить время первоначальной загрузки страницы на 40-60%, и обеспечить более комфортный просмотр web-карты.

3.4 Конечный результат

На рисунках 3.6, 3.7, 3.8, 3.9 представлены результаты работы разработанного картографического сервиса. Работоспособность сервиса была проверена с наиболее популярными браузерами, такими как MS Internet Explorer, Mozilla Firefox, Google Chrome, Opera. Обеспечена оперативная обработка поступающей информации GRIB, и ее визуализация.

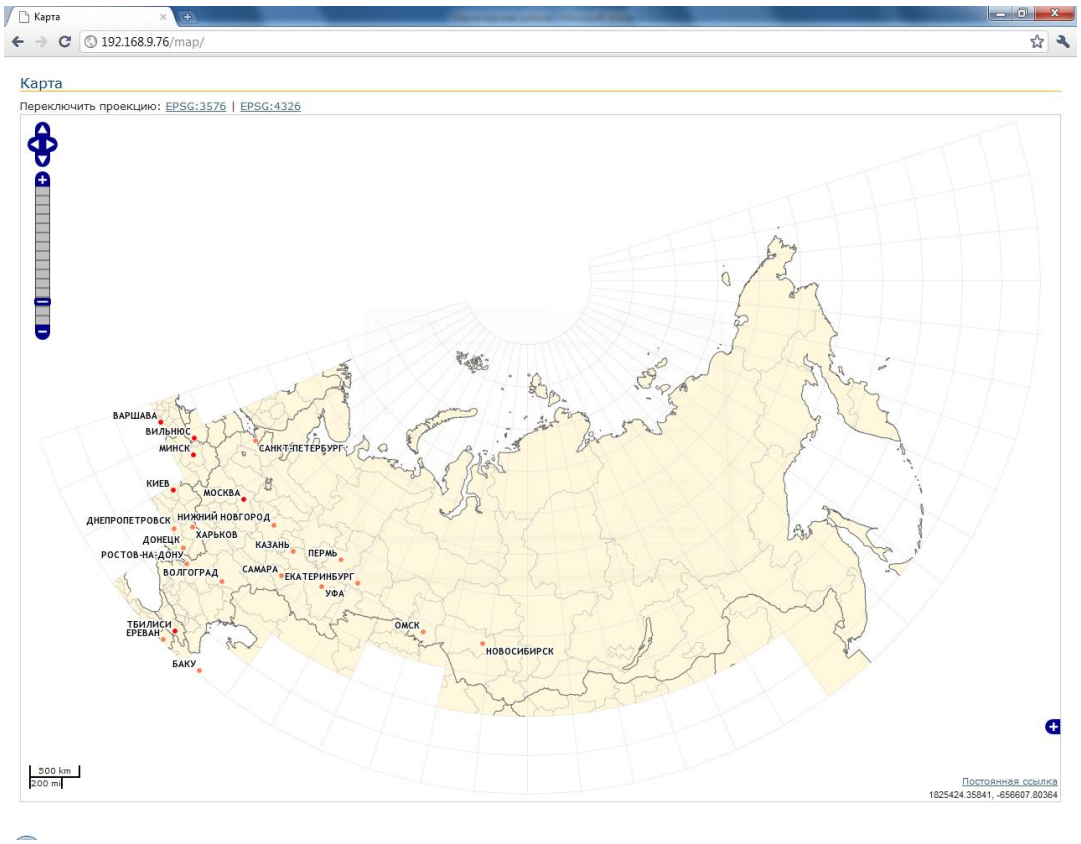


Рисунок 3.6 – Web-карта в проекции EPSG:3576

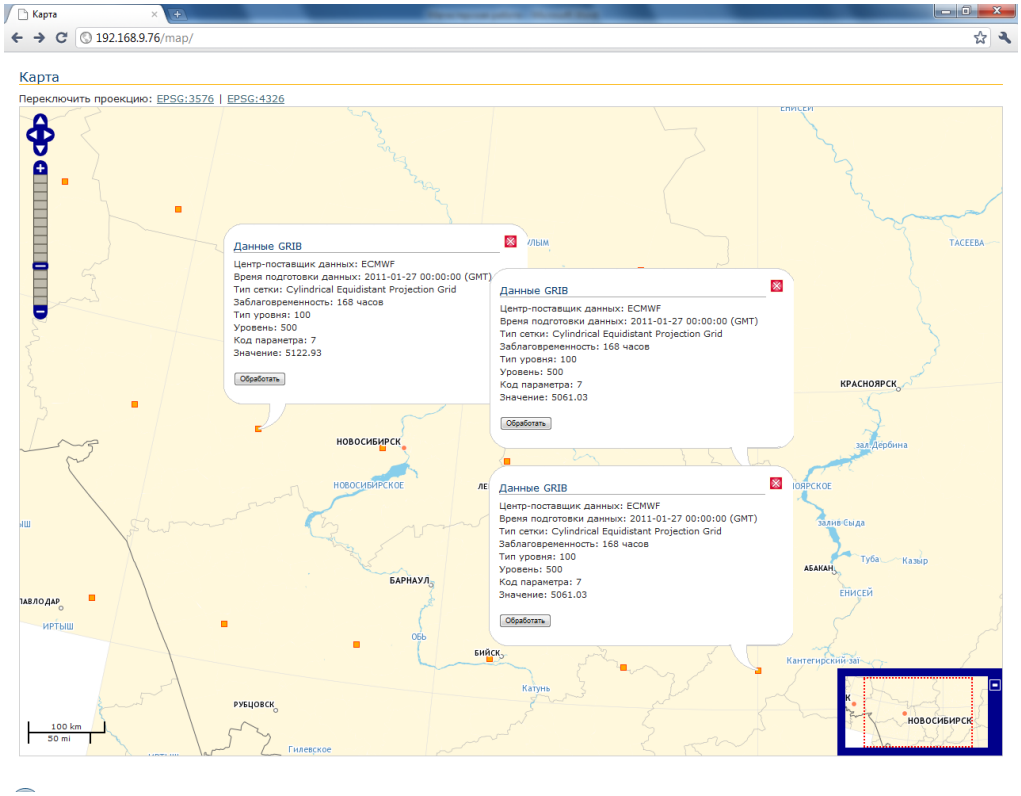


Рисунок 3.7 – Web-карта в проекции EPSG:3576, с атрибутивной информацией GRIB

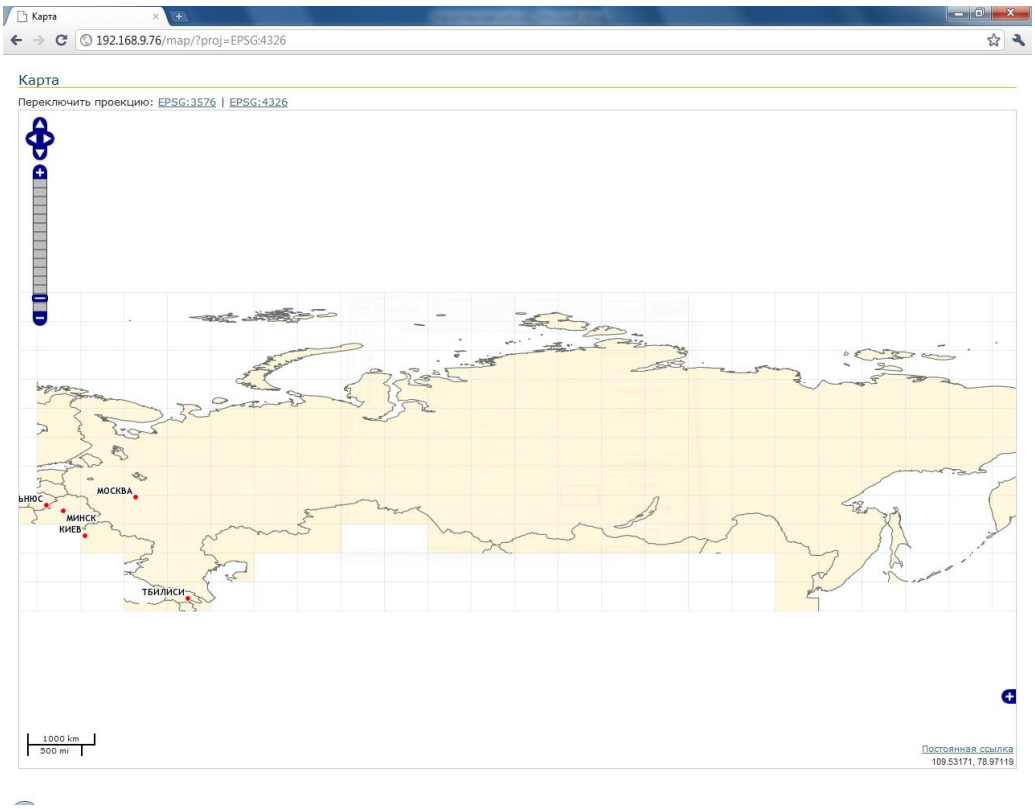


Рисунок 3.8 – Web-карта в проекции EPSG:4326

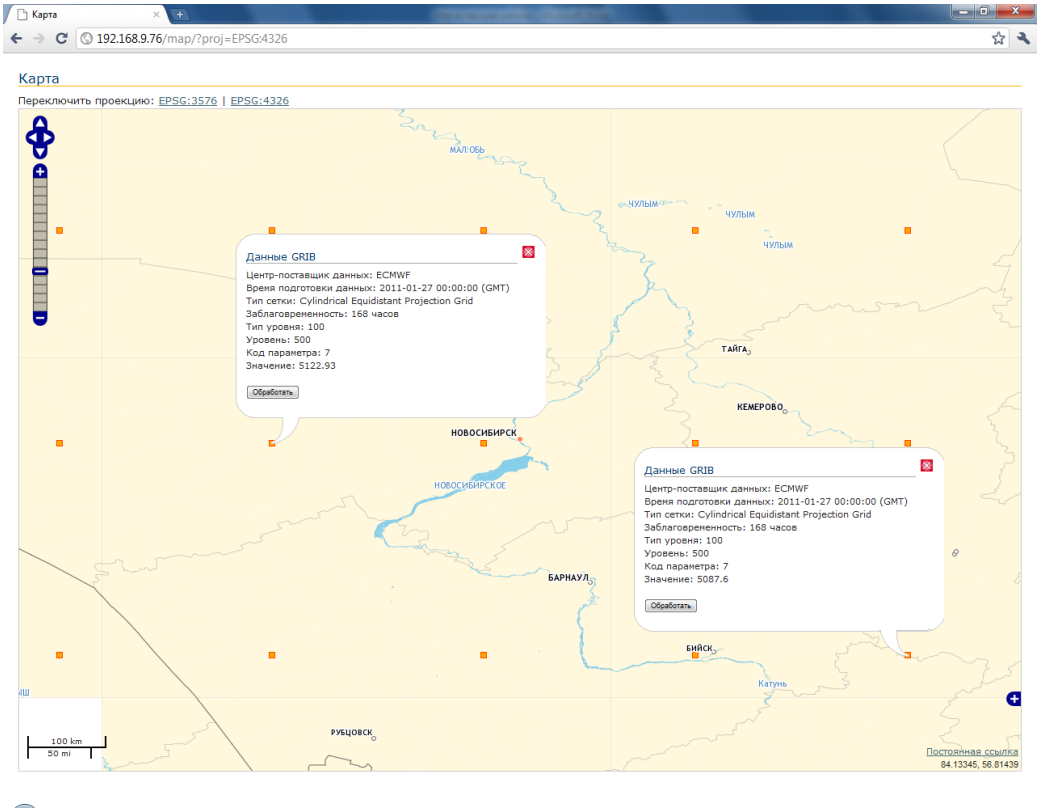


Рисунок 3.9 – Web-карта в проекции EPSG:4326, с атрибутивной информацией GRIB

Масштабирование данного сервиса можно обеспечить согласно следующей схеме, представленной на рисунке 3.10.

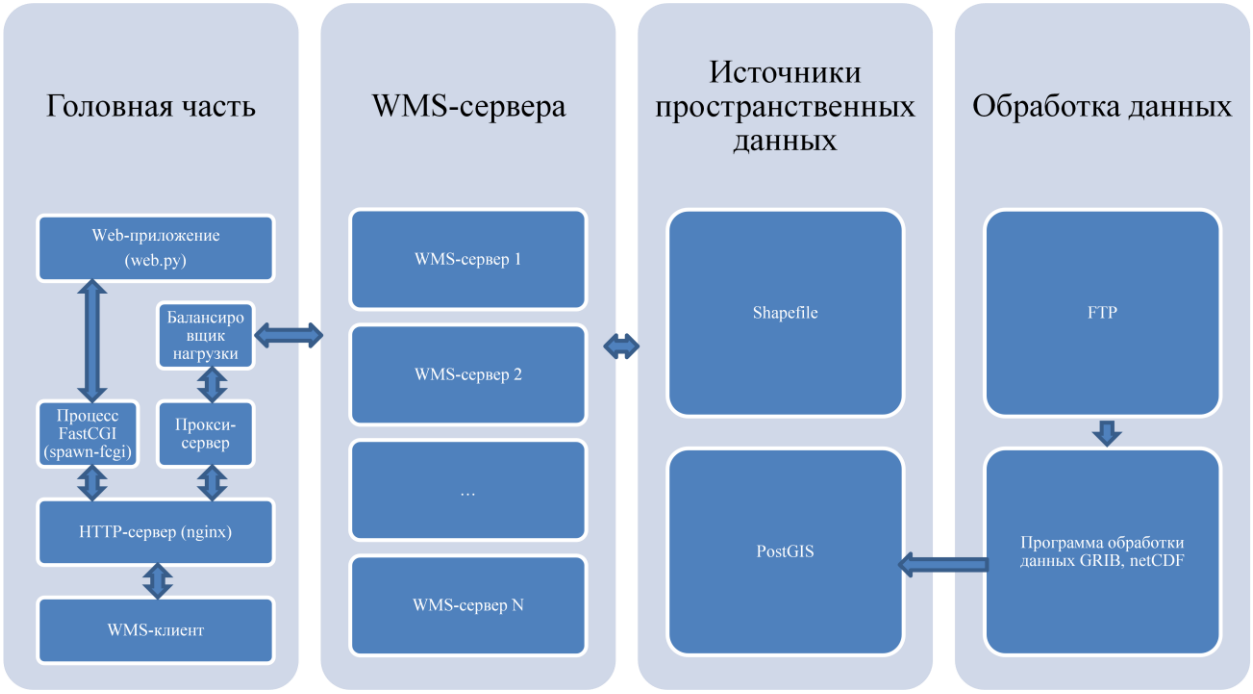


Рисунок 3.10 – Возможная реализация масштабируемости сервиса

Благодаря схеме, представленной выше, можно обеспечить большее число одновременно обрабатываемых запросов, за счет того, что генерацией картографического изображения будут заниматься несколько wms-серверов.

4 Внедрение технологии на сервере http://sibnigmi.ru

Разработанное программное обеспечение внедрено на сервере http://sibnigmi.ru как часть технологической цепочки обработки и визуализации данных-результатов расчетов прогностической модели COSMO. Разработанная система размещена в разделе «Продукция»

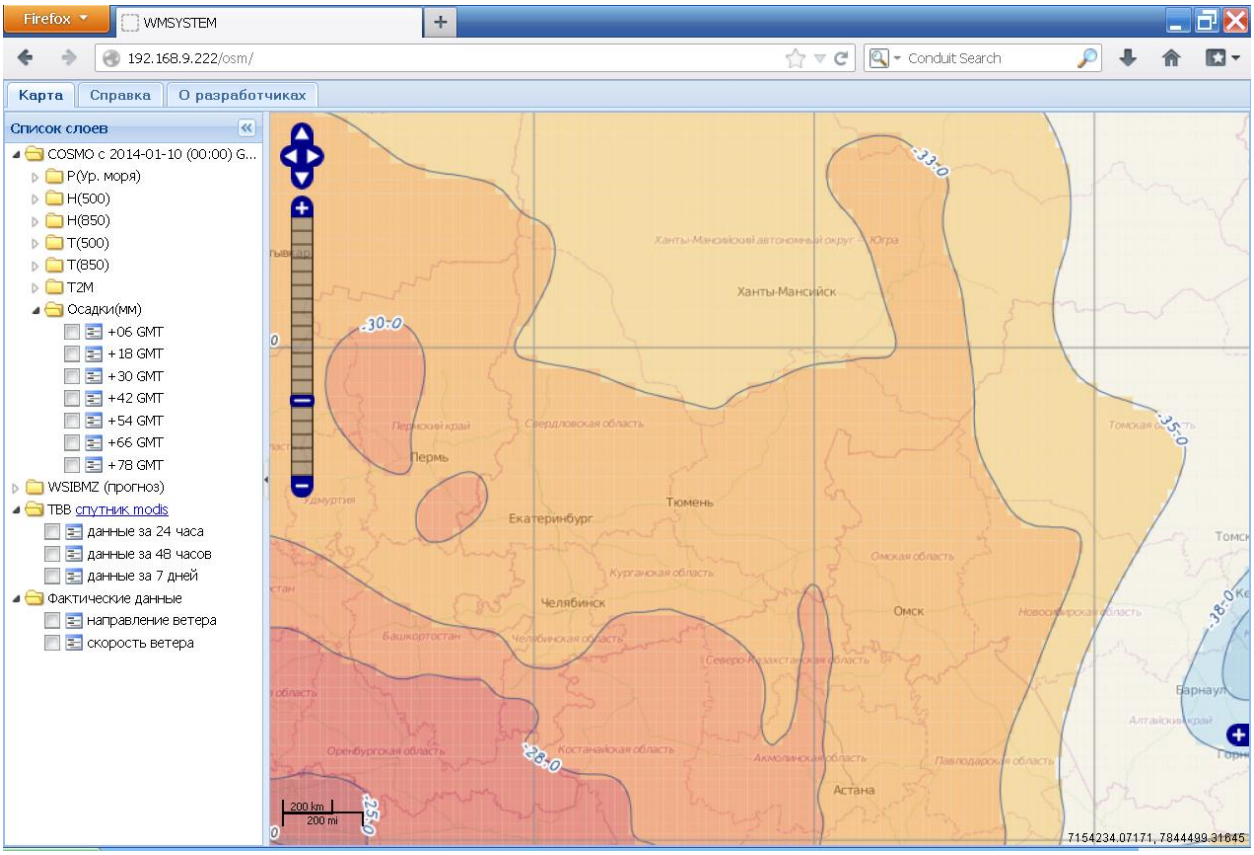


Рис. 4.1 – Пример интерфейса и данных, скриншот с сайта sibnigmi.ru

В панели слева доступны выбор слоев для визуализации. Реализованы 4 дерева слоев, соответсвующие техническому заданию для выполнения НИР:

- Дерево слоев COSMO (доступен выбор любой заблаговременности до 48 ч с шагом 3ч.:, представление в виде изолиний, изополос с заполнением цветом(с прозрачностью), изолиний с маркерами значений)
    - Поле давления (изолинии без закраски, с маркерами значений)

- Поле высоты геопотенциала 500 (изолинии без закраски, с маркерами значений)
- Поле высоты геопотенциала 800 (изолинии без закраски, с маркерами значений)
- Поле температуры на высоте геопотенциала 500 (изополосы, заполнение цветом)
- Поле температуры на высоте геопотенциала 850 (изополосы, заполнение цветом)
- Поле температуры приземной (изополосы, заполнение цветом)
- Осадки (зоны, заполнение цветом)

- Визуализация табличных данных физико-статистической схемы интепретации: прогноз WSIBMZ (отв. Исп. Здерева М.Я.)
  - Прогноз температуры;
  - Прогноз осадков;
  - Прогноз класса горимости с привязкой к классу Нестерова.
- Визуализация спутниковых данных – точки вероятного возгорания (спутник MODIS, NASA)
  - Термические точки за 24 часа
  - Термические точки за 48 часов
  - Термические точки за 7 суток
- Фактическся погода:
  - Направление ветра
  - Скорость ветра.

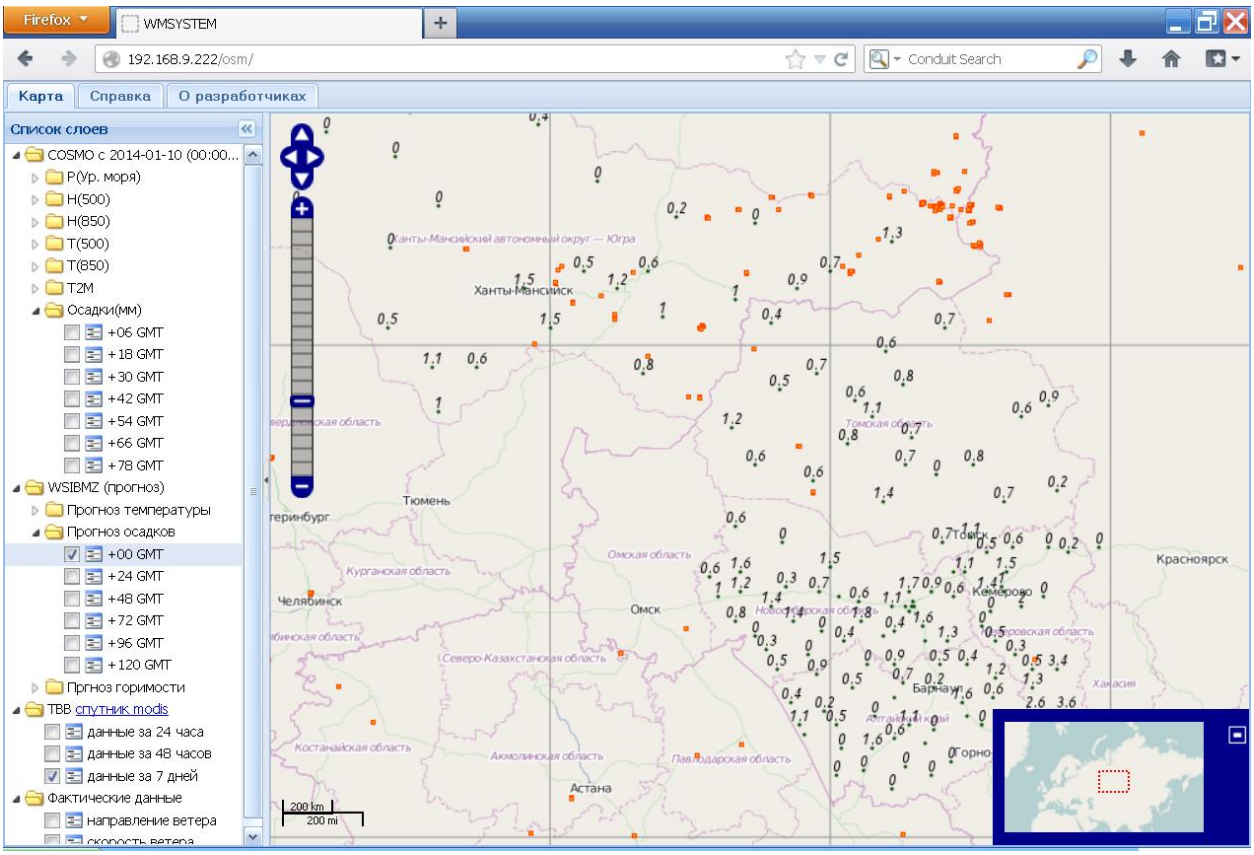


Рис. 4.2 – Пример интерфейса и данных, скреншот с сайта sibnigmi.ru

Интерактивное взаимодействие с пользователем реализовано путем вплывающего окна с отображением метеограммы (тема 1.7.46 2011- прервана в результате прекращения финансирования. Результаты темы, выполненные на тот момент, вошли в тему 1.1.1.4)

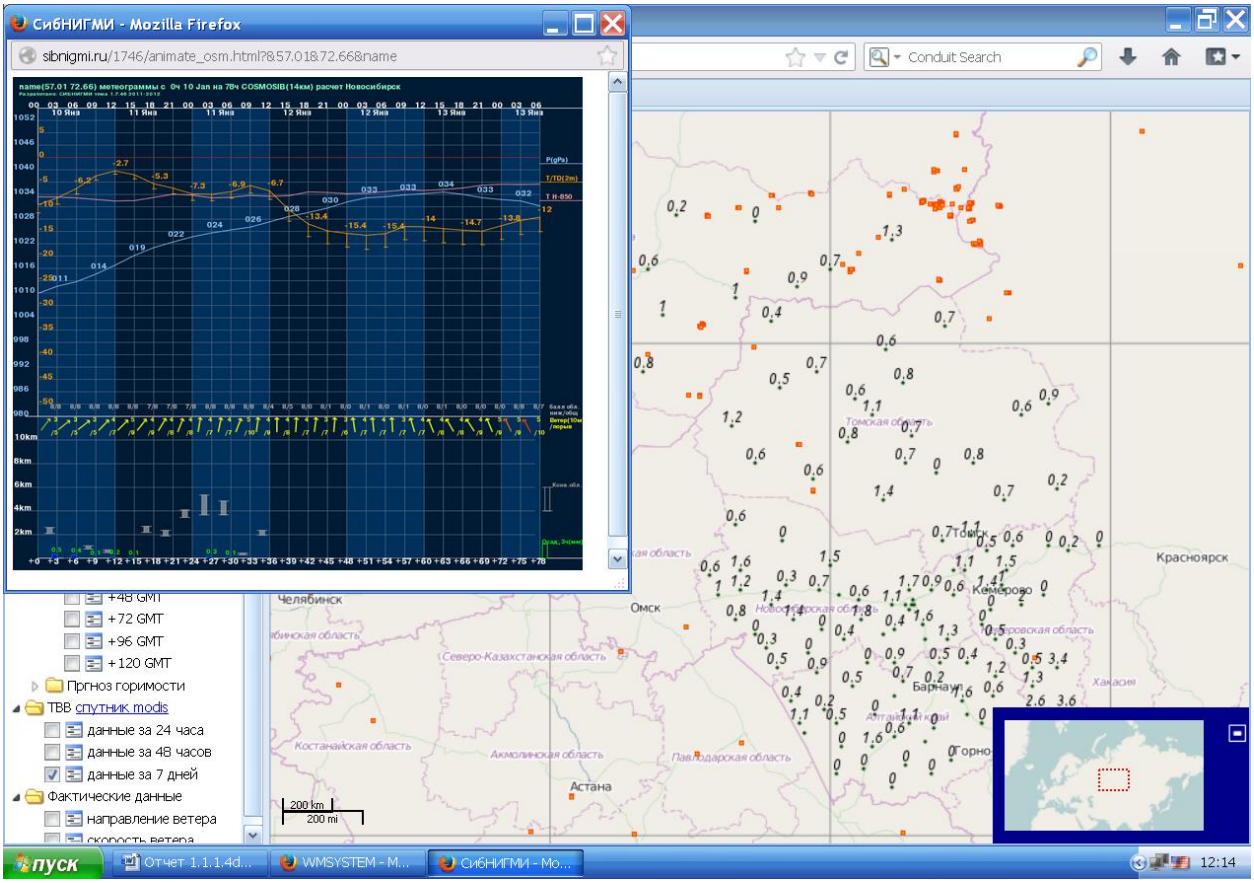


Рис.4.3. – Пример всплывающего окна.

На момент создания отчета сервер находится в режиме опытной эксплуатации в режиме круглосуточной работы.

ЗАКЛЮЧЕНИЕ

В данной работе была исследована эффективность отдельных компонентов, а также определены оптимальные характеристики для web-картографического сервиса. На основе этих данных была выполнена разработка комплексной информационной системы обработки и визуализации геофизических данных, Предложена схема масштабирования разработанного сервиса, что гарантирует дополнительную стабильность и производительность информационной системы в целом.

В соответсвии с Техническом Заданием, разработана технология многослойной векторной визуализации продукции мезомасштабных моделей высокого разрешения, построенная на базе свободно распространяемых компонент с открытым исходным кодом, внедренная в технологическую цепочку расчетов прогнозов погоды.

Достигнуты следующие показатели:

- Выступлений на конференциях 2

- Зарегистрированных программных продуктов: 2.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. OGC Standards and Specifications [Электронный ресурс]: Описание и спецификации стандартов OGC. – Режим доступа: http://www.opengeospatial.org/standards
2. A GUIDE TO THE CODE FORM FM 92-IX Ext. GRIB [Электронный ресурс]: Описание кода GRIB. – Режим доступа: http://www.wmo.int/pages/prog/www/WDM/Guides/Guide-binary-2.html
3. PROJ.4 - Cartographic Projections Library [Электронный ресурс]: Библиотека для работы с картографическими проекциями PROJ.4. – Режим доступа: http://trac.osgeo.org/proj/
4. Welcome to MapServer [Электронный ресурс]: Картографический сервер MapServer. – Режим доступа: http://mapserver.org/
5. GeoServer is an open source software server [Электронный ресурс]: Картографический сервер GeoServer. – Режим доступа: http://geoserver.org/display/GEOS/Welcome
6. Tyler Mitchell: Web Mapping Illustrated / Tyler Mitchell. – California: O'Reilly Media, 2005. – 368 с.
7. FOSS4G WMS Benchmark [Электронный ресурс]: Сравнение производительности WMS-серверов. – Режим доступа: http://wiki.osgeo.org/wiki/FOSS4G_Benchmark
8. PostGIS Documentation [Электронный ресурс]: Документация расширения PostGIS. – Режим доступа: http://postgis.refractions.net/documentation/
9. Beazley D. Python Essential Reference. – New York: Addison-Wesley, 2010. – 745 с.
10. Hazzard E. OpenLayers 2.10. – Birmingham: Packt Publishing Limited, 2011. – 372 с.
11. Web.py is a web framework for Python that is as simple as it is powerful [Электронный ресурс]: Руководство к библиотеке Web.py. – Режим доступа: http://webpy.org/cookbook/
12. HTTP-сервер Nginx [Электронный ресурс]: Документация. – Режим доступа: http://sysoev.ru/nginx/docs/
13. Clement Nedelcu: Nginx HTTP Server / Clement Nedelcu. – Birmingham: Packt Publishing Limited, 2011. – 348 с.

14. Саммерфилд М. Программирование на Python 3 / Перевод А. Киселева. – Санкт-Петербург: Символ-Плюс, 2009. – 608с.

15. Котов М.С. Разработка комплексной системы визуализации геофизических данных/ Магистерская диссертация, Новосибирск НГТУ 2011- 112с.