# Краткое описание решения

Решение представляет собой два проекта Visual Studio 2015 на основе .NET Framework 4.6:

1. Библиотека классов Geolocation.Data. Содержит функциональность загрузки данных из файла БД, кеширования данных и обработки запросов.
2. Веб-сайт ASP.NET MVC5 c WebAPI WebApp1. Является одностраничным приложением, написанным с использованием knockout, bootstrap и Sammy.

Для запуска проекта необходимо прописать две настройки в секции appSettings:

1. DatabaseFile (строка), путь до файла с БД.
2. DatabaseAvailableFast. True, если требуется минимизировать времени инициализации БД. False, если инициализация должна включать подготовку кеша данных.

Также в решении присутствует проект Geolocation.Test, в котором проводились тесты на корректность и производительность.

# Исходные положения

Поскольку описание задания представляет возможность для различного понимания некоторых аспектов решения, приведу положения, которые были приняты за основу при разработке.

1. Предполагается, что тип строковых данных sbyte[n] используется в качестве исходных данных для конструктура string(sbyte\*). Соответственно, все сроки гарантированно завершаются одним или более символами \0. Используется текущая системная кодировка, соответствующая System.Text.Encoding.Default. Следовательно, кодировка заранее не известна.
2. БД не будет использоваться ни в каких иных целях, кроме заявленных в описании задачи. Следовательно, кеш может хранить в себе итоговые ответы, готовые к отправке по сети в формате JSON и кодировке UTF-8. Сжатие GZip или deflate в данной версии не предусмотрено, но может быть внедрено в дальнейшем.
3. Базовые цели разработки – максимально быстрая загрузка файла БД и обслуживание существенного объёма запросов пользователей.
4. Невозможно сделать какие-либо предположения о распределении запросов.

# Проблемы при решении

Таблица IP-адресов содержит некорректные данные, начиная с записи №99038 (по счёту с 0). Здесь ip\_from больше ip\_to. Судя по всему, происходит переполнение возможных значений для 4-байтового беззнакового целого, что не позволяет использовать эту и следующие записи. Кроме того, данная ситуация противоречит условию задачи, в котором сказано, что индекс интервалов сортирован по ip\_from. Для информации привожу строковое представление данных в файле ip.txt и файл БД.

# Загрузка данных

Для загрузки используется отображение буфера byte[] на массивы структур с примитивными типами данных и массивами фиксированной длины. Есть три реализации загрузки, расположенные в порядке уменьшения времени загрузки:

1. С использованием MMF. Наиболее простая и наименее производительная реализация.
2. Полная буферизация файла БД и копирование его участков в 4 различных буфера. Более производительное решение, приводящее к фрагментации LOH.
3. Последовательное считывание файла в буфер размером 64кб (чтобы не фрагментировать LOH) и копирование его содержимого в 4 буфера. Более сложное и самое производительное решение.

Судя по замерам, БД загружается в память в течение 10-15мс и становится полностью готовой к обработке запросов.

# Кеширование данных

Кешируются готовые к отправке по сети ответы в формате JSON в кодировке UTF-8. Формируется кеш местоположений для ответа на запрос об IP-адресе и «кеш городов». Интервалы IP-адресов в ответ не включаются.

Всегда формируется полный кеш для всех возможных запросов: 100000 местоположений и 9340 городов. Кеш может быть сформирован в рамках процедуры инициализации БД, и тогда для этого используются все ресурсы процессора. В этом случае инициализация БД на тестовой машине занимает не более 200мс (i7, 4ядра, 8 потоков). Также кеш может формироваться после инициализации, когда уже могут поступать запросы от пользователей, и тогда формирование кеша занимает не более 25% ресурсов процессора, но не менее одного ядра.

Также записи в кеше сохраняются при обработке запросов от пользователей.

Доступ к кешу неблокирующий, в том числе в процессе его формирования. Используется реализация сериализатора в JSON, оптимизированная по производительности.

# Обработка запросов

При обработке запросов используются алгоритмы двоичного поиска по сортированным массивам.

Тестовые замеры, проведённые без использования клиентской инфраструктуры веб-приложения, показали, что обработка 1 млрд запросов на IP-адрес занимает не более 2 минут, 1 млрд запросов на город – не более 40 секунд.

# Дальнейшие пути оптимизации

Можно кешировать сжатые данные. Если в случае кеша местоположений это может не дать существенного эффекта, поскольку средняя длина ответа порядка 180 байт, то в случае кеша городов преимущество может быть ощутимым.

Можно попробовать реализовать кастомный двоичный поиск по интервалам IP.

Можно ускорить формирование кеша за счёт предварительной подготовки потоков.