Задание 2. Map reduce

Начало: 23 января 2017 года  
Конец: 5 февраля 2017 года (23:59:59)

# Цель задания

Целью лабораторной работы является моделирование системы удаленных вычислений, т.е. системы, предоставляющей свое машинное время для решения трудоемких задач по клиентским запросам. Имеются сервера, считающиеся системами с высокой вычислительной мощностью, и клиент, решающий некоторую задачу путем разделения ее на подзадачи, раздачи частей по серверам (map) и дальнейшего сокращения результатов (reduce).

В качестве решаемой задачи – решение задачи вычисления количества слов в тексте. Задача легко разделяется на более мелкие задачи

Используемые технологии: ввод – вывод из файлов, сетевое взаимодействие, многопоточность, сериализация, загрузчик классов, reflection

Требуется написать приложение типа client-server (WCF).

Приложение состоит из трех элементов: Master Server – менеджер вычислений и управления Worker’ами. Worker – приложения осуществляющие вычисления map/reduce в определенном контексте. Клиентское приложение – второе посылает Master Server’у сами функции Map и Reduce.

Сервер может поддерживать соединение одновременно с произвольным количеством клиентов, но одновременно исполняет задачи только одного клиента.

# Описание задания

**Требования к Master Server.** Сервер умеет регистрировать сервисы Workers двух типов – Map и Reduce. Так же сервер принимает функции Map и Reduce для Workers от Клиента. Следит за исполнением и очередями задача для Workers. Загрузка файлов для Workers может быть осуществлена так же через Master и он должен будет разделить файлы каждому из Workers. Master Server передает Workers функции в виде сериализованных объектов.

**Требования к Worker Servers.** Уметь регистрироваться на Master Server отправкой своих параметров. Так же исполнять функции Map/Reduce присланные с Master вместе с контекстом (файлом для обработки). Передавать Master Server свое состояние.

**Требования к Клиенту.** Передавать Master Серверу функции Map/Reduce в определенном формате (DataContract)

Все приложения должны быть консольными и уметь выводит весь лог работы.

# Map-Reduce

Алгоритм Map-Reduce не следует путать с одноименным фреймворком распределенных вычислений, хотя в определенном смысле целью лабораторной работы является построение упрощенной модели этого фреймворка.

Идея алгоритма состоит в том, что Master Server делит задачу на куски и рассылает Workers, которые генерируют ему ответы. Эти ответы Master Server склеивает в конечный результат.

Например, перед Master Server стоит задача найти все простые числа от одного до ста. Допустим, у Master Server имеется три подключенных сервера вычислений. Master Server делит задачу, например, на десять диапазонов по 10 чисел и рассылает серверам, собирая их ответы (шаг Map). Получившиеся ответы рассылаются северам Reduce для склеивания в конечный результат (шаг Reduce) и результат высылается Клиенту.

# Ввод и вывод

Ввод/вывод осуществляется через чтение стандартных для .NET потоков и файлов (Stream и System.IO)

# Установление сетевого подключения.

Сетевое подключение между двумя процессами устанавливается через протокол TCP или HTTP, через сервис WCF. Нужно иметь представление о механизм соединения WCF сервиса и понимать, что такое протокол SOAP и как стерилизуются объекты для передачи. Нужно знать отличия NetDataContractSerializer и DataContractSerializer.

# Serialization и Reflection

Каким образом мы можем взять произвольный поток байт, преобразовать его в класс, создать экземпляр этого класса и выполнить его методы? Эта задача решается с помощью технологий serialization и reflection

Serialization – отвечает за преобразование потока байт в определение класса. В случае, если класс хранится каким-либо нестандартным методом – например, лежит на сети, в виде xml, зашифрован, а также в любых других нестандартных вариантах, может потребоваться написание собственного загрузчика классов, способного к получению данных из нестандартного места. В нашем случае нам надо преобразовать только поток байт, пришедший по сети, и для этого мы можем использовать стандартный NetDataContractSerializer.

Reflection – В **информатике** **отражение** или **рефлексия** означает процесс, во время которого **программа** может отслеживать и модифицировать собственную структуру и поведение во время выполнения.

## Оценка

|  |  |
| --- | --- |
| Реализация клиент-серверной модели | **15 баллов** |

## Оформление

Вашу программу следует размести в Git (Желательно создать учетную Visual Studio Online (VSTS). Должно быть минимум два модуля правильно оформленных (Client и Server). Логика должна находится в отдельных классах и быть тестируемой. На вход программе должна быть возможность подавать данные с консоли. Клиенты должны писать все сообщения в лог файл (должно быт можно указать в командной строке), в том виде котором они приходят от сервера.

Весь код должен быть покрыт модульными тестами. Ссылку на доступ к репозиторию можно прислать вместо архива.

Структура:

**src** – исходники, без каких либо бинарных файлов

**lib** – дополнительные библиотеки, которые используются в вашем решении (кроме стандартных). Если пользуетесь **.nuget**, то заливать библиотеки не надо

**bin** – в этой директории должен хранится zip архив с вашими исполняемыми файлами и библиотеками. После распаковки архива, программа должна запускаться без дополнительных действий. Если есть особенности, указывайте их в readme.txt

Не забудьте положить в архив файл **readme.txt**. В файле описать интерфейс программы (алгоритм работы с программой, пункты меню, управляющие клавиши)

!Внимание: обратите внимание на правильное именование и структуру работы, а также на содержание файла readme.txt. Если ваша работа не будет соответствовать требованиям, баллы могут быть снижены. Время сдачи работы оценивается по последнему комиту. Если у вас есть модификации к работе после последнего срока – делайте branch и заливайте изменения туда.

# Результаты работы

Результаты смотрите в интернете. Совместная работа или обмен кусками кода, при условии что факт командной работы не был указан в readme.txt, считается плагиатом и оценивается в -5 баллов всем участникам.

# Дополнительный материал

* <https://msdn.microsoft.com/en-us/library/system.net.sockets%28v=vs.110%29.aspx>
* <http://www.java2s.com/Code/CSharp/Network/SimpleTcpServer.htm>
* <http://www.codeproject.com/Articles/13232/A-very-basic-TCP-server-written-in-C>
* <http://www.codeproject.com/Articles/488668/Csharp-TCP-Server>