Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Разработка среды для облачных вычислений

Курсовая работа студента 345 группы

Чуновкина Фёдора Дмитриевича

Научный руководитель ……. А.В. Бондарев

аспирант кафедры системного программирования

Санкт-Петербург

2010

Оглавление

Введение

Стремительное развитие информационных технологий за последние несколько десятков лет приучили людей к удобству вычислительной техники, которая сейчас используется повсеместно. Однако вычислительные ресурсы стоят денег, и, столкнувшись с очередной задачей, требующей применения подобных благ цивилизации, современный человек обладает целым рядом альтернативных возможностей.

Наиболее распространённым и традиционным решением в кругу конечных потребителей является использование локальных вычислительных мощностей – приобретение в личную собственность технических средств и использование их по-своему усмотрению для различных задач. Из очевидных минусов подобного подхода – малые мощности, которых рано или поздно начнёт не хватать для интересующего класса вычислительных задач.

Другим подходом является использование вычислительных мощностей на удалённых машинах - мощных серверах, обслуживающих тысячи пользователей единовременно. Такой подход более удобен неискушенному пользователю, так как освобождает его от необходимости самому иметь дело с аппаратными средствами, но по-прежнему не защищён от перегрузок самих серверов ввиду ограниченности их мощностей.

В последнее время широкое распространение получили так называемые «облачные вычисления» - предоставление пользователю абстрактных вычислительных мощностей, которые физически распределены на многих удалённых устройствах, образующих так называемое «облако». Для пользователя данный подход значительно удобнее предыдущего, так как он платит не за покупку или пользование конкретными аппаратными средства, а лишь за непосредственно выполненные для него расчёты.

Отдельного внимания заслуживают ставшие активно развиваться в конце прошлого тысячелетия проекты распределённых вычислений. В отличие от рассмотренных выше моделей, в которых конечный пользователей (клиент) организовывал или арендовал вычислительные мощности для своих задач, в этих проектах заинтересованное в вычислениях лицо устанавливает сервер и привлекает людей поучаствовать (чаще всего практически на безвозмездной основе) в решении поставленной задачи. Сервер раздаёт подключающимся клиентам-исполнителям фрагменты общей задачи и потом собирает результаты проведённых ими вычислений. Появление такого рода проектов обусловлено предположением, что большую часть времени компьютеры простых пользователей простаивают.

Каждый из рассмотренных подходов требует наличия под него специального программного обеспечения, в котором прослеживаются две противоположные тенденции: предоставление программного обеспечения как продукта или как услуги. Сейчас уже с уверенностью можно сказать, что второй подход стремительно завоёвывает все более стойкую позицию.

Рассматривая вышеперечисленные модели использования вычислительных ресурсов, можно проследить две кардинально противоположные проблемы:

- Большинство вычислительных ресурсов простаивает.
- Для ряда задач одиночных ресурсов часто бывает недостаточно.

В данной курсовой работе была рассмотрена попытка решения этих проблем путём объединения идей проектов распределённых вычислений и облачной архитектуры.

1. Постановка задачи

В рамках курсовой работы была поставлена задача объединить идеи проектов распределённых вычислений с облачной архитектурой. Конкретно было предложено создать среду, которую легко можно было бы развернуть на потребительских вычислительных мощностях, объединив их, таким образом, в «облако», позволяющее проводить в себе различные вычислительные расчеты. От облачной архитектуры данная среда заимствует уровень абстракции аппаратных средств от конечных пользователей, то есть пользователь не знает, где именно будет обработан его запрос на вычисления. В то же время предполагается реализовать возможность поддержания среды вычислительными средствами самих пользователей, по аналогии с проектами распределённых вычислений.

Конкретными задачами, поставленными передо мной, были:

- Написание прототипа вышеописанной среды, обладающего примитивными элементами самоуправления и распределения ресурсов.
- Демонстрация среды в действии на примере некоторой выбранной предметной области.
- Оценка перспективности предложенной идеи.

2. Реализация

2.1. Концепция

Среда (система) проектировалась из расчёта на максимальную гибкость и расширяемость. Увеличить вычислительную мощность можно простейшим подключением любого персонального компьютера или сервера (в стадии прототипа было решено ограничиться использованием операционной системы Microsoft Windows с установленным .NET Framework).

Среда предоставляет открытые интерфейсы для создания различных клиентских приложений, а также для расширения возможностей функциональных блоков самой системы. Посредством библиотек-плагинов (разработкой которых могут заниматься сторонние производители) возможно включить в систему поддержку различных физических устройств хранения данных, систем управления базами данных, а также наделить систему любой интересующей вычислительной функциональностью.

В качестве клиентов системы могут выступать сторонние приложения под любые платформы и устройства, имеющие доступ к сети и поддерживающие HTTP-протокол передачи данных. Вокруг системы возможно построение различных окружений, реализующих удобный пользовательский интерфейс к вычислительным мощностям. При этом система не завязана на одно единственное окружение - один экземпляр работающей системы может обслуживать множества различных окружений с различными клиентскими приложениями. Это позволяет на базе развёрнутой среды предоставлять различные комплексные услуги пользователям.

Для обеспечения универсальности упор сделан на поддержку массового аппаратного обеспечения. В отличие от продуктов, работающих с профессиональным оборудованием, данная среда не может полагаться на надёжность и защищённость используемых вычислительных ресурсов, а поэтому реализует поддержку этих качеств на программном уровне.

2.2. Архитектура

Среда представляет собой распределённую совокупность веб-сервисов шести различных типов, взаимодействующих друг с другом посредством обычного TCP/IP стека (рис. 1). Запущенная система сама управляет добавлением и удалением сервисов, в соответствии с

количеством поступающих запросов и своими нуждами. Каждый из типов сервисов может присутствовать в системе в любом количестве экземпляров.

Сердцем системы являются сервисы типа Cloud Controller (в дальнейшем CC), управляющие ресурсами системы и следящие за её целостностью и правильным функционированием. Для повышения надёжности системы возможно создание множества CC-сервисов (как дублирующих, так и образующих древовидную структуру). CC-сервисы управляют созданием, восстановлением и удалением всех сервисов в системе (в том числе других CC-сервисов).

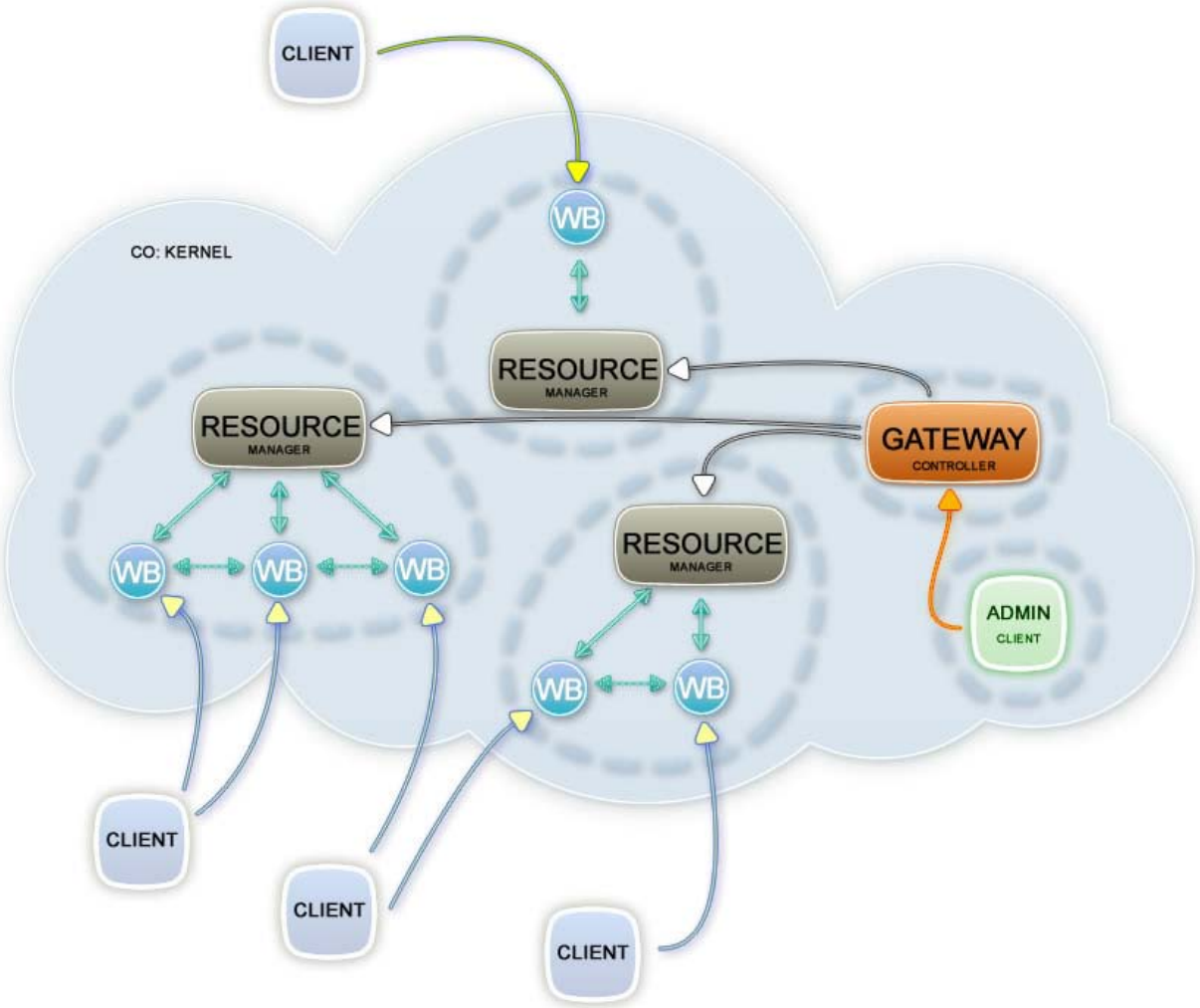

Рисунок 1. Архитектура среды

На каждой машине (персональном компьютере или сервере), подключаемой к системе (в том числе на машине, инициировавшей создание экземпляра системы), запускается сервис типа Resource Manager (RM), управляющий ресурсами данной конкретной машины. Одна машина может быть подключена сразу к нескольким экземплярам системы. В отличие от CC-сервисов, которые привязаны к конкретному экземпляру системы, RM-сервис руководит распределением ресурсов конкретной машины между всеми системами, к

которым она подключена. Таким образом, на каждой машине может присутствовать не более одного RM-сервиса, а во всей системе количество RM-сервисов будет совпадать с количеством подключённых машин.

Для журналирования работы системы, хранения информации о текущем состоянии, а также структурированных пользовательских данных используются сервисы типа Database Manager (DM), которые обеспечивают доступ к конкретным системам управлениям базами данных. Расширяемость DM-сервисов подключением дополнительных библиотек-плагинов позволяет потенциально использовать в системе любые СУБД и обеспечить при этом единый интерфейс доступа к ним.

Хранение информации в системе обеспечивается сервисами типа Storage Manager (SM), которые управляют конкретными физическими хранилищами данных. Расширяемость SM-сервисов подключением дополнительных библиотек-плагинов позволяет потенциально использовать в системе любые физические хранилища данных и обеспечить при этом единый интерфейс доступа к ним.

Непосредственные вычисления внутри системы осуществляются сервисами типа Work Block (WB). Функциональность WB-сервисов расширяется подключением дополнительных библиотек-плагинов. Всем клиентским приложениям для выполнения необходимых им операций системой выделяется один или более WB-сервисов, причём дальнейшее общение между клиентами и WB-сервисами происходит напрямую. Сама система также может использовать WB-сервисы для своих вычислительных задач.

Подключение к системе клиентских приложений происходит через сервисы типа Gateway (GW), адреса которых представляют собой входные точки в систему. GW-сервисы реализуют единый стандартизированный интерфейс для общения клиентов с системой. Используя этот интерфейс, наряду с интерфейсом к WB-сервисам, сторонние производители могут писать свои собственные клиентские приложения для системы.

2.3. Выбор технологии

В свете обширности идеи и отсутствия достаточного количества времени и ресурсов, на этапе создания прототипа было решено ограничиться готовыми технологиями организации сетевого взаимодействия, а конкретно использованием технологии WCF (Windows Communication Foundation), входящей в Microsoft .NET Framework. Это сузило круг машин, которые могут использоваться для поддержания работы системы, до

имеющих операционную систему Microsoft Window с установленным .NET Framework 3.5, что, однако, предполагается достаточным для оценки перспективности идеи (хотя также рассматривалась возможность оптимизации кода прототипа системы для запуска отдельных компонент под Mono).

2.4. Особенности предметной области

Хотя ключевой целью работы являлось написание непосредственно среды для построения распределённых вычислительных систем, трудно было бы демонстрировать систему в действии без выбора некой предметной области, в которой (в качестве примера) можно было бы показать достигнутые результаты.

На сегодняшний день на рынке всё большее место занимает обработка и хранение мультимедиа содержимого. Например, видеохостинги, видеоконференции, системы видеонаблюдения, и т.д. Объём видеоданных в мире очень быстро растёт, растут и затраты на их хранение и обработку. Применение в данной сфере предложенных идей распределённых облачных вычислений выглядит хорошей и перспективной задумкой.

Подобный выбор предметной области привёл к необходимости изучения и использования некоторых дополнительных технологий, в частности DirectShow, на базе которой реализованы почти все мультимедийные составляющие операционной системы Windows.

2.5. Проблемы и их решения

Несмотря на предполагаемую простоту, на первых стадиях работы возникали проблемы с использованием технологии WCF. Много усилий было потрачено на изучение разнообразных типов привязок (binding), определяющих используемые для сетевого взаимодействия протоколы. В конце концов, выбор был сделан в пользу наиболее простого и самого распространённого типа привязки (так называемый BasicHttpBinding), работающего поверх обычных POST и GET запросов протокола HTTP. Как оказалось, многие заманчивые возможности WCF крайне слабо поддержаны сторонними производителями. Так, создание простейшего RIA (Rich Internet Application)-клиента уже становилось совсем не тривиальной задачей с каким-либо другим типом привязки (рассматривалось использование таких популярных технологий как Adobe Flash и Microsoft Silverlight).

Следующая проблема встала при первых шагах в выбранную предметную область, и заключалась в передаче непрерывных потоков мультимедиа данных. Направленность WCF на общение при помощи сообщений оказалась очень неудобна в данной ситуации. Первым решением был переход на использование протокола HTTP вручную, которое сразу же решило проблему передачи данных от облака (в нашем случае – некоторого WB-сервиса) к клиентам системы, но оставила открытым вопрос вещания данных с клиента (например, с веб-камеры) в облако. Следующим действием был шаг на ещё более низкий уровень – использование лишь транспортного протокола TCP с ручной реализацией логики общения поверх установленного соединения. Это решение окончательно разрешило проблему. Было решено оставить обратную совместимость передачи данных от облака к клиенту по протоколу HTTP, что позволило проигрывать входящие потоки стандартными средствами самих клиентов.

Наконец, следующей осознанной проблемой было соединение компьютеров, находящихся за преобразователями сетевых адресов (NAT, Network Address Translator). Хотя некоторые исследования в данном направлении были проведены (протокол STUN), данный вопрос остался до конца не разрешенным.

Заключение

Результатом проделанной работы стал прототип программного продукта, позволяющий развернуть среду распределённых облачных вычислений. Среда запускается на любом компьютере с операционной системой Microsoft Windows и установленным .NET Framework. Поддерживается подключение неограниченного количества дополнительных удалённых компьютеров-помощников. Управляющие блоки среды обладают примитивными способностями к распределению нагрузки между доступными вычислительными ресурсами.

Работоспособность запущенной среды продемонстрирована на примере обработки мультимедиа-информации. Реализована возможность захвата аудио-видео потоков (со сжатием в форматы MP3 и FLV соответственно) клиентами и трансляции их через облако другим клиентам. Подобный функционал позволяет построить, например, распределённый сервис видеоконференций в облаке – достаточно лишь снабдить запущенную среду неким окружением (веб-сервером), предоставляющим соответствующий пользовательский интерфейс. Примитивное окружение для среды также было создано, однако его описание выходит за рамки данной работы.

Предложенная идея показала себя вполне жизнеспособной, однако выбор технологий в угоду скорости разработки несколько смазал общее впечатление. Использование высокоуровневых технологий, таких как WCF, приводит к весьма существенным затратам производительности. Первостепенной задачей по дальнейшей работе над проектом видится перенос логики, продемонстрированной прототипом, на более быстрые низкоуровневые технологии.

Список литературы

1. Википедия – Свободная энциклопедия
   http://ru.wikipedia.org/
2. FLV File Format Specification
   http://www.adobe.com/devnet/flv/pdf/video_file_format_spec_v10.pdf
3. Microsoft Developer Network (MSDN)
   http://msdn.microsoft.com/