СЕРВЕРНАЯ ПЛАТФОРМА RUNSERVER

Краткое описание возможностей и технологий

Платформа RunServer является инструментарием и ядром (middleware) для создания серверов приложений реального времени, в том числе, онлайн игр (Massive Multiplayer Online Game). RunServer построен на основе .Net Framework, что подразумевает работу сервера под ОС Windows для достижения максимального быстродействия. Использование современных технологий позволяет достичь высокой производительности платформы со сравнительно небольшими затратами на аппаратное обеспечение.

Курсивом выделены возможности, которые находятся в разработке.

Сетевая подсистема

- в качестве сетевого транспорта могут использоваться следующие протоколы:
  - поточно-ориентированный протокол TCP;
  - *протокол UDP*;
  - IPC (Inter-Process Communications);
- возможно использование IPC в любой части сервера для обмена данными с другими приложениями посредством Shared Memory;
- отдельный C++ модуль с использованием Input-Output Completion Ports (IOCP);
- унифицированная система заданий, обрабатывающая сетевые данные в едином конвейере с задачами сервера;
- обработка клиентских пакетов в различных потоках в зависимости от типа потока и приоритета пакета;
- высокий уровень абстракции, позволяющий использовать любой формат принимаемых и отсылаемых данных, в т.ч. с шифрованием (полным или частичным) в реальном времени;
- идентификация пользователей по уникальному хэш-коду адреса;
- менеджер активных подключений, построенный на не-блокирующих коллекциях, исключающих многопоточные блокировки при подключении и отключении пользователей;
- многопоточная обработка принятых и отправленных данных;
- отсылка keep-alive сообщений (поддержание связи при отсутствии сетевого трафика);
- замер входящего и исходящего трафика;
- использование как Managed, так и Unmanaged памяти из системного Heap;
- возможность работы в кластерном режиме с единым шлюзом (концентратором трафика) и обмена данными между другими серверами кластера;

Модуль БД

- объектно-ориентированная система представления данных, позволяющая работать с записями БД как с обьектами и их коллекциями (более быстрый и удобный аналог коммерческих решений вроде NHibernate [1]);
- возможность задания индексных полей любого типа;
- работа с массивами данных (группировка столбцов в массив);
- работа с вложенными объектами (группировка столбцов в отдельные суб-объекты или их массивы);
- задание значений по-умолчанию в коде;
- работа с бинарными данными в любой БД через формат base64 или напрямую;
- возможность связывания различных типов объектов (один к одному, один ко многим, многие ко многим);
- автоматическая загрузка связанных объектов;
- возможность каскадного удаления связанных объектов;
- автоматическая генерация вспомогательного кода, выполняющего загрузку и выгрузку данных в объекты;
- отслеживание изменений в объекте для инкрементных обновлений и гарантированной записи только изменившихся данных;
- интрузивный подсчет ссылок для сбора статистической информации;
- возможность автоматической инкрементальной нумерации объектов (собственная реализация Incrementing Primary Key исключающая возможность появления объектов с одинаковыми идентификаторами на всех серверах кластера);
- возможность загрузки связанных объектов при помощи хранимых процедур;
- возможность постраничной загрузки объектов;
- данные каждой таблицы могут быть кешированы одним из трех методов:
    - хранение всех объектов БД или их части в массиве для доступа по идентификатору (индексу). Дает максимальную скорость загрузки данных таблиц, у которых идентификаторы расположены последовательно;
    - хранение объектов в хеш-таблице. Дает возможность быстрого доступа к данным таблиц, у которых идентификаторы имеют большой разброс;
    - хранение слабых ссылок (Weak Reference) на объекты, что позволяет кешировать объект пока задействован в программе и высвобождать его, когда необходима дополнительная память и объект более не используется;
- система отложенных операций с использованием транзакций на уровене БД и собственных отложенных транзакций;
- возможность поиска данных различными методами:
    - доступ к объекту по идентификатору;
    - доступ к группе объектов по списку идентификаторов. Если объекты группы находятся в кеше – будут возвращены ссылки на их локальные копии;
    - доступ к группе объектов по SQL запросу. Запрос выполняется БД и возвращает список идентификаторов, по которым выбирается из кеша или БД результат;
    - доступ к объекту или группе объектов по индексному столбцу;

    - возможность сканирования БД для извлечения содержимого индексных столбцов без загрузки всех данных;
- для одних данных в БД всегда будет создан один и только один объект, что предотвращает загрузку не актуальных данных, появление дубликатов;
- возможность извлечения базовых данных о таблице (количество записей, максимальный идентификатор и пр.);
- автоматическое создание таблиц в БД в случае их отсутствия;
- возможность экспорта данных из одной БД в другую, в т.ч. с трансформациями, разворачиванием base64 полей и пр.;
- собственный модуль соединения с БД (не основан на ADO .NET) позволяющий работать со следующими типами БД:
    - MYSQL [2];
    - Microsoft SQL Server [3];
    - Oracle TimesTen [4] (используется собственный C++ модуль для подключения [5]);
    - Oracle Database [6];
    - Firebird [7];
    - ANTs Data Server [8];
    - SQLite Database [9];
    - ODBC database connection;
    - OLEDB database connection;
    - XML data storage (хранение данных в XML файлах, совместимых с внутренним форматом ADO .NET DataSet);
    - Remote DB connection (соединение с внешней программой, выполняющей роль кеширующего сервера);
    - SQL output only (формирование SQL кода вместо операций добавления, изменения и удаления объектов);

Ядро

- многопоточный конвейер, позволяющий выполнение задач в одной из следующих сред:
    - собственный Thread Pool с фиксированными параметрами и количеством потоков выполнения, базирующийся на IOCP;
    - стандартный Thread Pool .Net;
    - вспомогательные потоки с различными приоритетами и задержками выполнения;
    - вызывающий поток;

- сбор статистики о времени выполнения каждой задачи, средней загрузке потока, пулов, системы вцелом;
- система высокоточных таймеров (High Precision Timers);
- восемь различных приоритетов с различным поведением при обработке;
- выполнение таймеров единоразово, с указанным интервалом, с возмостями воскрешения и перезапуска;
- автоматическая смена очереди выполнения при приближении времени срабатывания;
- система блокировок, исключающая ложные срабатывания;
- возможность установки различных событий OnStart, OnTick, OnStop и др.;

- динамическая загрузка, компиляция и перезагрузка внешних скриптов на всех поддерживаемых языках .Net;
- возможность загрузки заранее откомпилированных скриптов из внешних DLL файлов;
- автоматический вызов статических инициализаторов и финализаторов при загрузке и выгрузке скриптов;
- набор собственных коллекций, которые превосходят по скорости и выделению памяти коллекции .Net за счет узкой специализации:
    - LinkedArrayList – список, использующий связанные массивы (Linked Arrays) для хранения данных и наращивания без копирования;
    - LinkedQueue, FastLockQueue – FIFO (First In – First Out) очереди, основанные на связанных массивах;
    - LinkedStack – FILO (First In – Last Out) очередь, основанная на связанных массивах (стек);
    - SynchronizedDictionary – словарь, использующий ReaderWriterLock для синхронизации работы с данными;
    - SimpleListDictionary – словарь, использующий списки для хранения данных;
    - PooledList – список, использующий пул массивов объектов вместо создания новых массивов;
    - SortedLinkedList – сортированый связанный список;
    - StackList – коллекция реализованная в виде структуры и создающаяся в стеке;
    - CheckList – immutable коллекция, позволяющая быстро определять наличие или отсутствие элемента;
    - коллекции, представляющие собой структуры с различным числом элементов;
- собственные утилиты для облегчения и ускорения работы с коллекциями;
- собственные классы для работы с потоками данных;
- утилиты для работы с неуправляемой (Unmanaged) памятью;
- различные утилиты для задач синхронизации;
- менеджер логирования, осуществляющий перехват консольных сообщений и логирование их с различными уровнями детализации;
- работа в 32-битном и в 64-битном режиме;
- архивация потоков данных с помощью библиотек zlib [10] или SharpZipLib [11];
- собственный математический модуль;

Дополнительные наработки для MMOG

Система карт (Grid-Based Map System)

Система карт, которая тут описывается, не имеет ничего общего с ландшафтом или обходом препятствий. Под словом "карта" подразумевается область мира, в которой возможны какие-либо действия игрока и в которой могут находиться объекты.

Суть системы состоит в том, что игровой мир покрывается сеткой с большим количество квадратных ячеек, так называемых тайлов. Каждый тайл взаимодействует только с 8ю окружающими тайлами. Поле видимости и взаимодействия для игрока, находящегося в тайле - текущий тайл и 8 окружающих. Для удобства все 9 тайлов вынесены в специальную коллекцию и почти все действия с тайлами выполняются над всей коллекцией. Например, если отсылается информация о движении какого-то объекта, то ее получат все игроки в текущем и в окружающих тайлах.

Если игрок пишет сообщение, то его будут слышать только обитатели окружающих тайлов, которые находятся на определенном расстоянии от него. Проверять остальные тайлы нет смысла. Если операция затрагивает объекты на расстоянии меньше половины тайла (разговор на небольшом расстоянии, взаимодействие с объектом), используется оптимизация и обрабатывается только 4 тайла (текущий и три смежных).

Перемещение объекта между тайлами происходит с обязательной отсылкой пакетов удаления обитателям тех тайлов, которые больше не будут смежными с текущим и отсылкой пакетов создания обитателям тех, которые стали смежными с текущим. Пакеты отсылаются как о движущемся объекте, так и ему обо всех обитателях тайлов (если данный объект умеет обрабатывать такую информацию, т.е. является игроком).

Таким образом, мы отказываемся полностью от использования глобальных коллекций игроков, не игровых персонажей, объектов, заменяя их локальными коллекциями для каждого тайла. Исключение составляет коллекция игроков, которая нужна для глобальных действий вроде поиска, списков онлайна и подобных операций, но эта коллекция не используется для взаимодействия с окружающими объектами.

Система тайлов позволяет значительно уменьшить нагрузку на сервер за счет того, что активность во всех тайлах, где нет игроков, может быть отключена. Например, агрессивные компьютерные персонажи не сканируют на наличие игроков в пустых тайлах, не ходят и т.п.

Такая система карт позволяет кроме всего прочего работать с любым количеством миров, организовывать переходы между ними, обрабатывать динамические триггеры (ловушки, порталы) и т.п.

Дальнейшее развитие такой системы подразумевает автоматическое создание (спаун) игровых объектов и их удаление при полной неактивности. Максимальная реализация позволяет кроме этого создавать любое количество клонов миров (различных измерений) и навигацию по ним с автоматической очисткой неиспользуемого измерения.

Система коллективных действий

Специфика MMOG подразумевает наличие не управляемых персонажей с высоким уровнем социальных отношений - торговцев, охранников, жителей городов и т.п. Для обеспечения реалистичности поведения таких персонажей была создана система коллективных действий. На данном этапе система состоит из двух частей – подсистемы коллективных боевых действий и подсистемы коллективного передвижения. Подсистема боевых действий предполагает различные роли персонажей в бою – все персонажи делятся на атакующих, поддерживающих и дальнобойных. Каждый из них действует в соответствии со своей ролью. Подсистема коллективного передвижения позволяет организовывать движение групп персонажей в различных формациях, которые описываются отдельными формулами и позволяют создавать группы персонажей, которые выстраиваются колонной, шеренгой, кольцом, клином или произвольной фигурой. Лидер группы в свою очередь может являться членом другой группы, что позволяет создавать произвольные построения.

Регулировка трафика в зависимости от его значимости

В современных MMOG часто случаются ситуации, когда игрок получает информацию о каком-то объекте или другом игроке, которая для него не актуальна. Например, в местах массового скопления – городах, магазинах, массовых битвах – игроку не всегда интересно знать, какой именно предмет создал другой игрок, находящийся на пределе видимости. Также зачастую в массовых битвах детализация движений противников и союзников, не замешанных напрямую в сражении игрока, имеет небольшой приоритет, но в различных MMOG передается полностью, что может значительно ухудшить качество игры. Для оптимизации подобного трафика и реализации массовых битв была разработана система адаптивной регулировки трафика. Вокруг игрока описывается сложная фигура,

представляющая собой правильную полусферу перед игроком и деформированную полусферу (с уменьшенным радиусом по одной из осей) за спиной игрока. Все объекты, попадающие в область, ограниченную данной фигурой, попадают в список «объектов интереса» (Objects of Interest). Также в этот список попадает выбранная цель, текущие противники (те, с кем ведутся боевые действия в данный момент) и возможные союзники (как игроки, так и не-игровые персонажи). Данные о действиях объектов этого списка передаются игроку полностью, в то время как действия всех остальных объектов проходят через специальные фильтры. Например, большое количество пакетов, описывающих сложную траекторию движения, будет заменено несколькими пакетами с упрощенной фигурой, вспомогательные пакеты действий могут быть высланы в заархивированном виде и или отброшены.

Дальнейшим развитием этой системы будет добавление весовых коэффициентов к каждому пакету, а также формирование для каждого игрока некого значения «уровня детализации», который будет изменяться в зависимости от нагрузки. Это позволит адаптивно регулировать трафик и сохранять латентность и качественные ощущения от процесса игры на одном уровне в любой игровой ситуации.

Сноски:

[1] NHibernate:
http://www.hibernate.org/343.html
[2] MYSQL relational database:
http://www.mysql.com
[3] Microsoft SQL Server 2000:
http://www.microsoft.com/sql/default.mspx
[4] TimesTen In-Memory Database:
http://www.oracle.com/timesten
[5] TTProvider:
http://sourceforge.net/projects/ttprovider
[6] Oracle Database:
http://www.oracle.com/database/index.html
[7] Firebird:
http://www.firebirdsql.org
[8] ANTS High performance RDBMS:
http://www.ants.com/index.php?option=com_content&task=view&id=212&Itemid=4
[9] SQLite Database:
http://www.sqlite.org
[10] zlib:
http://www.zlib.net/
[11] SharpZipLib:
http://www.icsharpcode.net/OpenSource/SharpZipLib/