Нижегородский государственный университет им. Н.И. Лобачевского
Факультет вычислительной математики и кибернетики
Учебно-исследовательская лаборатория
”Информационные технологии”

Alchemi .NET Framework

Нижний Новгород
2007 г.

Содержание

Введение и основные концепции

В данном разделе описано, каким образом инструментарий Alchemi, разрабатываемый в университете Мельбурна, реализует концепции грид-компьютинга. Данный инструментарий интересен тем, что для его реализации была выбрана платформа Microsoft .NET и операционная система Microsoft Windows, тогда как большинство существующих инструментариев предназначены для работы в Unix. Таким образом, Alchemi является убедительным примером демонстрации возможностей платформы Microsoft .NET в области грид-компьютинга.

В настоящее время инструментарий Alchemi является проектом с открытым исходным кодом и доступен для свободного скачивания с сайта SourceForge.

Сеть как один компьютер

Основная идея *метакомпьютинга* состоит в использовании независимых компьютеров, объединенных в сеть, как будто они являются одной большой параллельной машиной или виртуальным суперкомпьютером. Эта идея имеет ряд преимуществ, к которым относится, например, низкая стоимость суммарных вычислительных ресурсов.

В то время как традиционные виртуальные машины были разработаны для небольшого числа гомогенных сильно связанных ресурсов, быстрый рост сети Интернет позволяет применять эти концепции в намного большем масштабе. К тому же, настольные ПК в корпоративных и домашних условиях не сильно нагружены – обычно используется только одна десятая часть мощности. Поэтому возникает интерес к использованию вычислительных мощностей, которые доступны в виде *свободных* циклов работы центрального процессора. Эта новая парадигма стала одной из идей *грид*.

Сегодня наблюдается быстро растущий интерес к технологиям грид со стороны коммерческих предприятий. Операционная система Microsoft Windows широко используется большинством организаций. Для более широкого распространения грид не хватает развития грид-платформ, поддерживающих операционную систему Microsoft Windows. Эта поддержка позволила бы эффективно использовать свободную вычислительную мощность настольных компьютеров и рабочих станций посредством создания виртуального вычислительного ресурса, стоимость которого значительно ниже стоимости традиционных суперкомпьютеров. В настоящее время практически отсутствует сервисно-ориентированное промежуточное программное обеспечение, поддерживающее Microsoft Windows. Инструментарий Alchemi, основанный на технологии Microsoft.NET, принадлежит к числу таких проектов.

Несмотря на то, что понятие грид может быть дано на интуитивно простом уровне, практическая реализация такой инфраструктуры сталкивается с множеством трудностей. Основные проблемы связаны с разнородностью, надежностью, разработкой приложений, планированием, управлением ресурсами и безопасностью. Технология Microsoft.NET может быть использована для решения всех перечисленных проблем. Отметим, в частности, поддержку удаленного исполнения (.NET Remoting и веб-сервисы), многопоточности, безопасности, асинхронного программирования, удаленного доступа к данным, управления исполнением и поддержку нескольких языков программирования. Все это делает инструментарий Microsoft.NET удобной платформой для разработки промежуточного программного обеспечения грид.

Принцип работы Alchemi

В вычислительной грид-сети, построенной на базе инструментария Alchemi, участвуют следующие четыре компоненты: *Менеджер* (Manager), *Исполнитель* (Executor), *Пользователь* (User) и *Межплатформенный менеджер* (Cross-Platform Manager). Взаимосвязи этих сущностей показаны на рис. 1.

Как видно из представленной схемы, инструментарий Alchemi построен по клиент-серверной схеме. Для объединения некоторой совокупности компьютеров в вычислительную сеть, нужно выбрать один узел, который будет играть роль сервера, и установить на него Менеджер. На один или несколько компьютеров сети устанавливаются клиенты – Исполнители, которые настраиваются на работу с Менеджером. В дистрибутив Alchemi входят удобные программы инсталляции, требующие минимальной настройки, так что процесс создания Alchemi-грид является достаточно простым.

Исполнитель поддерживает два режима работы с Менеджером: *выделенный* (dedicated) и *невыделенный* (non-dedicated). Первый режим означает, что Менеджер отдает поток на выполнение незамедлительно, тогда как во втором случае выполнение потока инициируется самим Исполнителем. В

невыделенном режиме Исполнители могут работать через брандмауэры и NAT серверы, так как в этом случае создается лишь односторонняя связь с Менеджером. Выделенный режим лучше подходит для работы в локальной сети, в то время как невыделенный режим предпочтителен для работы в Интернет.

Пользователи могут разрабатывать, выполнять и осуществлять мониторинг грид-приложений, используя специальный .NET API и инструментальные средства, включенные в Alchemi SDK.

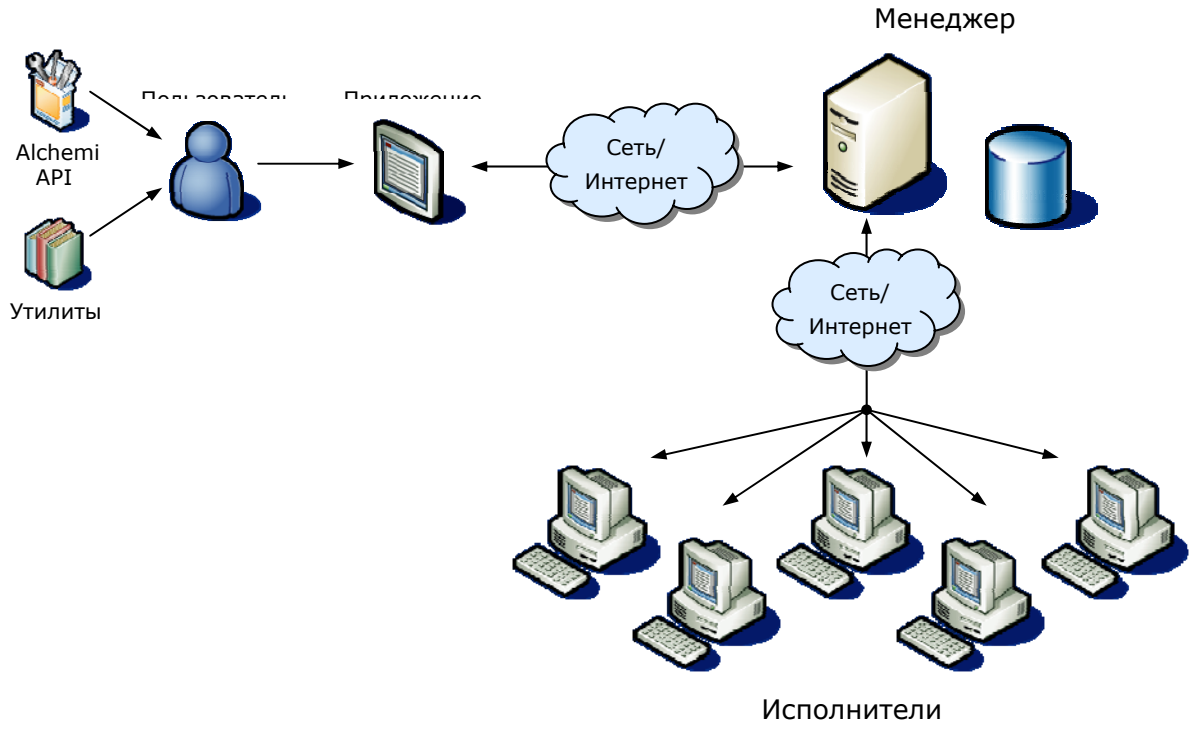

Рис. 1. Схема работы инструментария Alchemi

Инструментарий предлагает две модели программирования: модель *грид-потоков*, которая позволяет очень просто разработать новое приложение, а также модель *грид-заданий* для поддержки уже созданных программ.

Дополнительная компонента (не показана на рисунке) – *Межплатформенный менеджер* (*Cross-Platform Manager*) в виде веб-сервиса – предоставляет функциональную совместимость с другим *промежуточным* программным обеспечением (middleware) грид. Это означает, что грид-сеть, построенная на базе Alchemi, может входить в состав другой, более крупной грид-сети.

Возможные применения инструментария

Компоненты, входящие в состав инструментария Alchemi, могут использоваться для построения различных конфигураций вычислительной сети: *кластер* (desktop grid), *мультикластер* (multi-cluster grid), *глобальная грид-сеть* (global grid).

Кластер

Простейший сценарий применения инструментария Alchemi – создание вычислительного кластера из настольных компьютеров. В этом случае используется один Менеджер и несколько Исполнителей, сконфигурированных на работу с Менеджером. Схема такой вычислительной сети показана на рис. 2.

Рис. 2. Развертывание кластера

Один или несколько пользователей могут исполнять свои задания на кластере, подключаясь к единственному Менеджеру. Подобную схему можно применять как в локальных сетях, так и в Интернете. Однако эту схему вряд ли можно назвать надежной: если с Менеджером произойдет сбой, то вся вычислительная сеть окажется нерабочей.

Мультикластер

При такой организации вычислительной сети несколько Менеджеров объединяются в иерархическую структуру. Так же, как и в предыдущем случае, несколько Исполнителей и пользователей могут подключаться к Менеджеру на любом уровне иерархии (рис. 3).

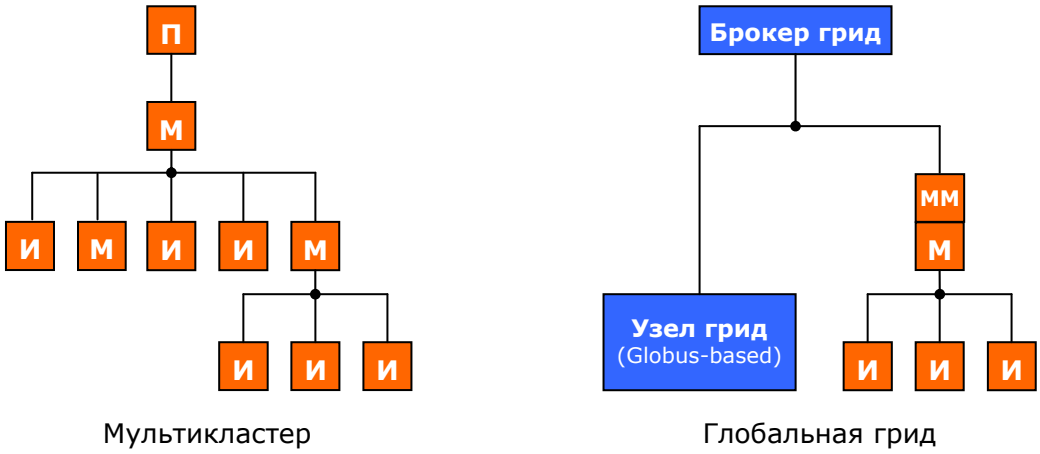

Рис. 3. Развертывание мультикластера и интеграция в глобальную грид

Отметим ключевой момент в этой схеме. Менеджер на каждом уровне иерархии, за исключением самого верхнего, конфигурируется на работу с вышестоящим Менеджером в качестве “промежуточного”. Для вышестоящего Менеджера он выглядит как Исполнитель. Таким образом, Менеджер реализует интерфейс Исполнителя. Промежуточный Менеджер играет двоякую роль: с одной стороны он выполняет функции Исполнителя (для вышестоящего Менеджера), с другой – функции Менеджера (для нижестоящих Исполнителей и Менеджеров). Подобная схема лучше подходит для использования в сети Интернет и обладает большей надежностью по сравнению с предыдущим вариантом.

После того, как пользователь отправляет грид-приложение на выполнение главному Менеджеру, последний получает некоторое число грид-потоков, ожидающих своего исполнения в грид-сети. По-умолчанию все грид-потоки имеют наивысший приоритет и начинают исполняться на доступных Исполнителях. Некоторые Исполнители могут оказаться промежуточными Менеджерами. В этом случае после получения потока от вышестоящего Менеджера, его приоритет понижается на единицу, и поток планируется на выполнение на локальных Исполнителях промежуточного Менеджера. Понижение приоритета приводит к тому, что поток перемещается вниз по иерархии Менеджеров. Таким образом, чем “ближе” поток находится к Исполнителю, тем с большим приоритетом он начинает выполняться. Это позволяет часть Alchemi-грид, находящуюся в пределах одного административного домена, разделять с другими организациями, обеспечивая совместную работу без воздействия на конечных пользователей.

Промежуточный Менеджер, как и обычный Исполнитель, может быть сконфигурирован на работу в двух режимах: выделенном и невыделенном. Подобная возможность позволяет более гибко использовать ресурсы и обеспечивать работу при ограничениях сети.

Глобальная грид

Межплатформенный Менеджер может быть использован для конструирования сегмента грид, отвечающего классической схеме. Компонента промежуточного программного обеспечения, такая как брокер ресурсов, может использовать Межплатформенный Менеджер в виде веб-сервиса для исполнения межплатформенных приложений (описываемые файлами задания) на сегменте Alchemi (кластер или мультикластер). Например, Alchemi-грид может входить в состав более крупной грид-сети, построенной на базе Globus Toolkit.

Обзор компонент инструментария

Выше упоминалось, что в работе грид-сети и грид-приложений принимают участие четыре компоненты. Для построения вычислительной грид на один из узлов сети нужно установить Менеджер, а на остальные – Исполнители, сконфигурировав их на работу с Менеджером. Пользователи могут выполнять свои приложения в грид, подключаясь к Менеджеру. Дополнительная компонента, Межплатформенный Менеджер, обеспечивает интеграцию с другим программным обеспечением промежуточного уровня. В этом разделе подробно рассматриваются все четыре компоненты: Менеджер, Исполнитель, Пользователь и Межплатформенный Менеджер.

Менеджер

Любое приложение (*грид-приложение*), разработанное для инструментария Alchemi, представляет собой совокупность *грид-потоков*, которые концептуально очень похожи на “обычные” потоки.

Менеджер управляет выполнением грид-приложений и грид-потоков, из которых состоят грид-приложения. Исполнители регистрируются у Менеджера, который осуществляет непрерывный мониторинг их работы. Грид-потоки, полученные от Пользователя, помещаются в пул, и составляется график их выполнения на доступных Исполнителях. Для каждого потока задается приоритет. Это можно сделать явно при создании потока или его отправке, а можно оставить значение по-умолчанию. Планирование осуществляется на основе приоритетов и принципа “первым пришел – первым обслужен” (First Come First Served – FCFS). Исполнители возвращают Менеджеру завершенные грид-потоки, которые впоследствии собираются воедино соответствующими Пользователями. Следует заметить, что в состав инструментария входит Scheduling API, позволяющее создавать свои планировщики.

Для аутентификации и авторизации используется модель безопасности на основе ролей. Список разрешений, а также список групп (которые представляют собой набор разрешений) хранятся вместе с Менеджером. Для выполнения любого действия, требующего авторизации, Пользователь или программа должны предъявить имя пользователя и пароль, после чего Менеджер разрешает выполнение этого действия, только если данная учетная запись принадлежит группе, которая содержит нужное разрешение.

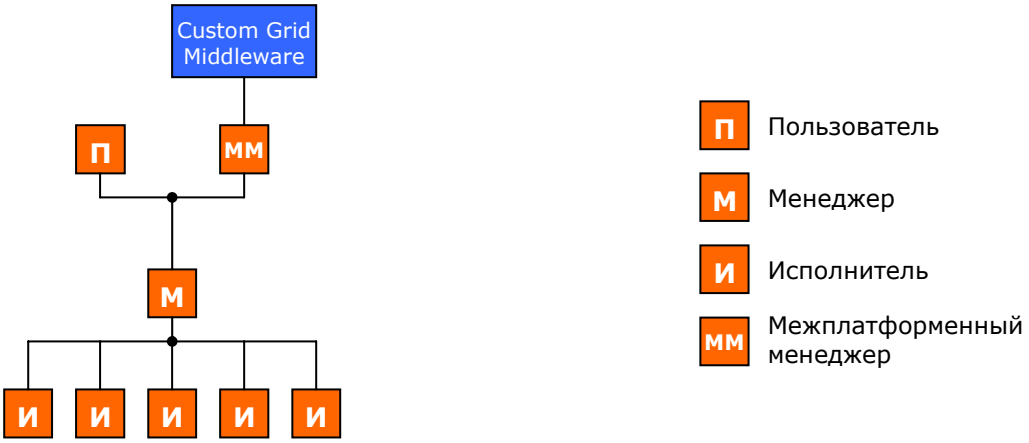

Рис. 4. Компоненты распределенной системы и их взаимосвязи

Как обсуждалось ранее, стабильность работы грид-системы играет ключевую роль в ее эффективности. Менеджер осуществляет непрерывный мониторинг работы Исполнителей. Грид-потоки, которые были запущены на отключившихся Исполнителях, перепланируются и запускаются вновь на доступных ресурсах (предполагается, что грид-потоки допускают возможность рестарта). В дополнение к этому все данные постоянно сохраняются на диск, так что в случае краха системы, Менеджер после повторного запуска не теряет своего состояния.

Исполнитель

Исполнитель получает грид-потоки от Менеджера и выполняет их. Исполнитель может работать в двух режимах. В *выделенном* режиме он представляет собой ресурс, полностью управляемый Менеджером. В *невыделенном* режиме Исполнитель выполняет грид-потоки во время простоя компьютера (когда запускается программа-заставка) или когда пользователь явно разрешает выполнение грид-потоков. В невыделенном режиме имеет место одностороннее взаимодействие между

Исполнителем и Менеджером, так как в этом случае Исполнитель сам посылает запрос на получение грид-потока для выполнения. Когда возможна двухсторонняя связь и используется выделенный режим, Менеджер взаимодействует с Исполнителем напрямую. В этом случае Менеджер явно указывает Исполнителю, какие потоки он должен выполнять. Таким образом, модель выполнения приложений, используемая в инструментарии Alchemi, позволяет получить двойную выгоду от:

- гибкого управления ресурсами, сочетая централизованное управление с принудительным исполнением и децентрализованное управление с программируемым планированием исполнения грид-потоков,
- гибкого развертывания системы при ограничениях сети, таких как proxy- и NAT-серверы, когда невозможна двусторонняя связь между Исполнителем и Менеджером.

Таким образом, можно сказать, что выделенный режим лучше подходит для локальных сетей, а невыделенный для сети Интернет.

Грид-потоки выполняются с набором разрешений `AlchemiGridThread`, которые настраиваются в рамках локальной политики безопасности .NET. Устанавливаемый набор разрешений определяет для грид-потоков (программный код которых является потенциально небезопасным) среду выполнения, ограниченную по ресурсам и безопасную для операционной системы.

Все потоки выполняются с наименьшим приоритетом, поэтому работа пользовательских приложений, имеющих более высокий приоритет, практически не затрагивается.

Пользователь

Грид-потоки выполняются на узлах Пользователей. Alchemi API скрывает реализацию грид от Пользователя и отвечает за выполнение необходимых служебных действий. К их числу относится отправление приложения и составляющих его грид-потоков на выполнение, оповещение Пользователя о завершившихся грид-потоках и получение результатов их работы, а также оповещение о неудачных грид-потоках и предоставление деталей ошибки.

Межплатформенный Менеджер

Межплатформенный Менеджер – это веб-сервис, который обладает частью функциональных возможностей Менеджера и позволяет Alchemi управлять выполнением грид-заданий (в противоположность грид-приложениям, использующим модель грид-потоков). Задания, отсылаемые Межплатформенному Менеджеру, преобразуются им в представление грид-потоков, с которым работает Менеджер. Затем грид-потоки планируются и выполняются обычным способом. Дополнительно Межплатформенный Менеджер обеспечивает совместимость с уже созданными грид-приложениями, позволяя использовать ресурсы Alchemi-грид на любой платформе, поддерживающей веб-сервисы.

На рис. 5 показана многоуровневая архитектурная схема инструментария Alchemi.

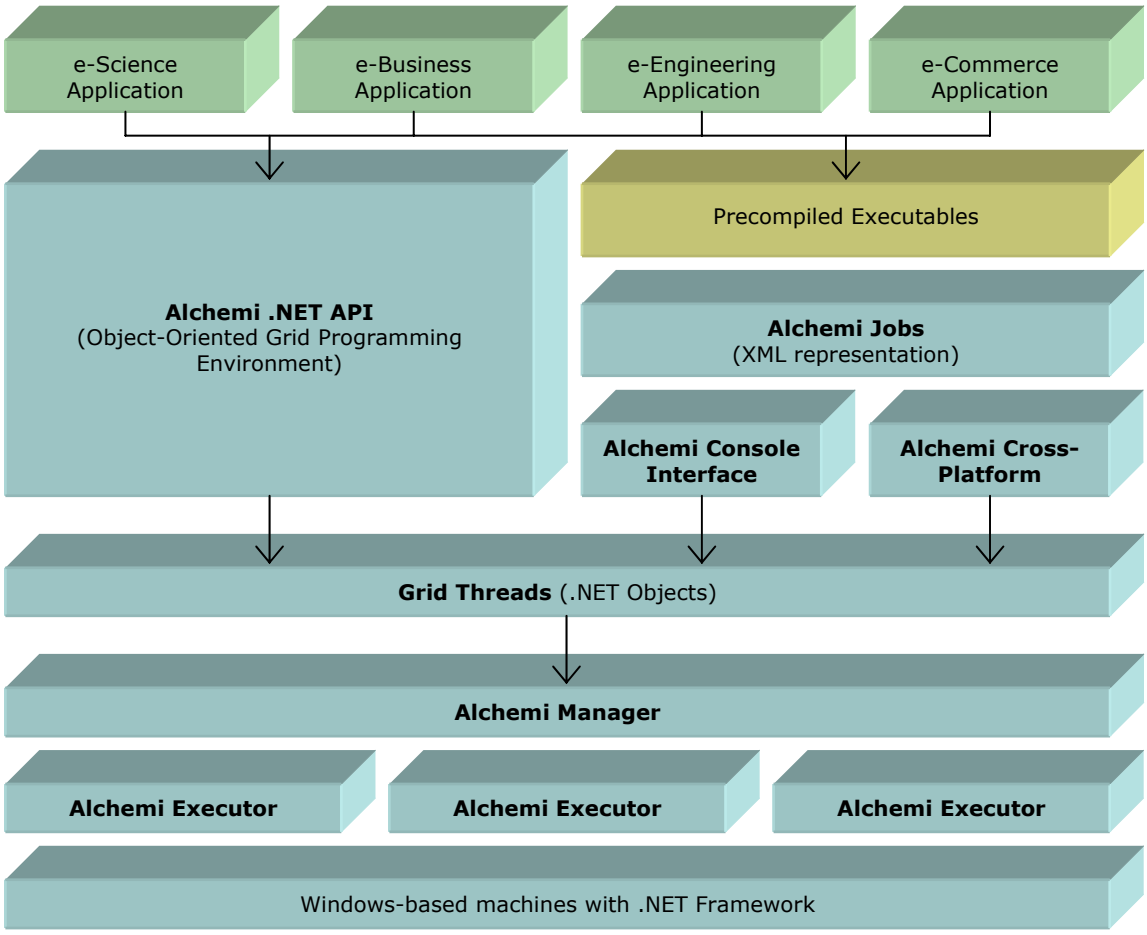

Рис. 5. Многоуровневая архитектурная схема инструментария Alchemi

Как уже отмечалось ранее, инструментарий Alchemi построен на основе парадигмы параллельного программирования “менеджер-исполнитель”, в которой центральная компонента системы организует обработку независимых единиц параллельного вычисления. В Alchemi этими независимыми единицами вычислительного процесса являются грид-потоки, содержащие инструкции, которые должны быть выполнены на одном грид-узле. Центральной компонентой, организующей процесс вычисления, является Менеджер.

Грид-приложение состоит из множества грид-потоков. Грид-приложения и грид-потоки создаются разработчиком как обычные .NET-объекты с помощью специального Alchemi .NET API. Когда приложение начинает свою работу, объекты грид-потоков отсылаются Менеджеру для исполнения в грид-сети на доступных Исполнителях.

Альтернативным вариантом являются грид-задания на основе файлов. Описание заданий хранится в XML-файлах специального формата. Грид-задания на основе файлов обеспечивают совместимость с существующими приложениями, для которых уже имеются откомпилированные исходные тексты. Задания могут отправляться посредством Консоли Alchemi (Alchemi Console Interface) или Межплатформенного Менеджера. XML-представление переводится в представление грид-потоков, которые затем отсылаются Менеджеру и исполняются обычным образом.

Концепции грид-приложений

Разработка грид-приложения может рассматриваться как написание распределенной программы для абстрактной “виртуальной” машины с множеством центральных процессоров. Основная задача различного *промежуточного* программного обеспечения состоит в том, чтобы предоставить программисту эту абстракцию. В различном программном обеспечении промежуточного уровня это

делается по-разному. В Alchemi выбран способ, упрощающий процесс разработки программного обеспечения для грид настолько, насколько это возможно.

Благодаря характеру инфраструктуры грид (слабосвязанные, гетерогенные ресурсы, взаимодействующие по ненадежному, с высокой латентностью каналу), грид-приложения должны обладать следующими особенностями:

- они могут быть распараллелены на множество *независимых* вычислительных частей,
- эти части в основном работают над вычислениями, а не над установкой соединения.

Как говорилось ранее, Alchemi поддерживает две модели для параллельных вычислений, которые подробнее рассматриваются ниже.

Модель грид-потоков

Инструментарий Alchemi позволяет достаточно быстро разрабатывать новые приложения для грид. Достигается это во многом благодаря объектно-ориентированной модели программирования, близкой к традиционному многопоточному программированию. Элементарной единицей параллельного исполнения в этом случае является *грид-поток*; множество грид-потоков образуют *грид-приложение*.

Разработчики, у которых нет необходимости досконально знать тонкости функционирования грид, могут полностью сконцентрироваться на основной логике и манипулировать такими абстракциями, как грид-приложение и грид-поток. Кроме того, данные абстракции позволяют использовать такие конструкции языка программирования, как события, применительно как к локальному, так и к удаленному коду. Вся эта функциональность доступна через Alchemi .NET API.

Дополнительным преимуществом этого подхода является то, что разработчики не ограничены только полностью параллельными программами, которые представляют собой множества *невзаимодействующих* грид-потоков. Возможно создание грид-приложений, в которых отдельные грид-потоки взаимодействуют между собой. Следует отметить, однако, что в настоящее время возможность взаимодействия грид-потоков еще не реализована, поддержка такой возможности планируется в будущем. Наконец, отметим, что программы, использующие Alchemi .NET API, могут быть написаны на любом .NET-языке, например, C#, VB.NET, Managed C++, J#, JScipt.NET.

Модель грид-заданий

Традиционные реализации грид-инфраструктуры предлагают высокоуровневую абстракцию “виртуальной машины”, в которой минимальной сущностью параллельного выполнения является *процесс*. Спецификация задания, которое должно быть выполнено в грид, состоит из списка входных файлов, выходных файлов и списка команд, которые должны быть выполнены.

Поддержка модели грид-заданий присутствует по следующим причинам:

- для поддержки существующих реализаций грид или не .NET-приложений;
- для функциональной совместимости с другим промежуточным программным обеспечением грид.

В дальнейшем будет рассматриваться лишь модель грид-потоков, освоение модели грид-заданий выходит за рамки данного учебного материала.

Сравнение концепций грид и многопроцессорности

Для того чтобы лучше разобраться с концепциями Alchemi-грид, можно представить определенную аналогию между грид и многопроцессорным компьютером.

Таблица 1. Сравнение концепций грид и многопроцессорности

Многопроцессорный компьютер	Грид
Операционная система	Alchemi
Процессор	Исполнитель
Высокоуровневые сервисы ОС Системный API	Alchemi .NET API
Низкоуровневые сервисы ОС	Менеджер
Процесс	Грид-приложение
Поток	Грид-поток

Разработка грид-приложения

Данный раздел дает введение в разработку приложений с использованием инструментария Alchemi. Предполагается, что читатель знаком со средой программирования Microsoft Visual Studio .NET и языком C#, которые используются для подготовки и компиляции приведенного ниже примера.

Описание задачи

В качестве примера будет разработано несложное грид-приложение для генерации простых чисел. Программа будет случайным образом генерировать большие числа и проверять в грид, какие из них являются простыми. Для проверки воспользуемся самым простым методом, который перебирает делители от единицы и до заданного числа. Если делителей оказалось два (единица и само число), то, по определению, число является простым. Возможная последовательная реализация подобного алгоритма может быть представлена следующим образом:

```
/// <summary>
/// Данная функция генерирует <code>count</code> целых числел и
/// последовательно проверяет, какие из них являются простыми.
/// </summary>
/// <param name="count">количество генерируемых чисел</param>
public void Generate(int count)
{
    Random random = new Random();

    for (int i = 0; i < count; i++)
    {
        int candidate = random.Next(1000000);

        if (CheckFactors(candidate) == 2)
        Console.WriteLine("Число {0} является простым", candidate);
    }
}

/// <summary>
/// Данный метод вычисляет количество делителей целого числа
/// <code>candidate</code>.
/// </summary>
/// <param name="candidate">входное целое число</param>
/// <returns>чило делителей</returns>
public int CheckFactors(int candidate)
{
    // Начальное число делителей полагаем равным нулю
    int factors = 0;

    // Поиск всевозможных делителей числа
    for (int i = 1; i <= candidate; i++)
    {
        if (0 == candidate % i) factors++;
    }

    return factors;
}
```

Безусловно, существуют гораздо более эффективные алгоритмы генерации простых чисел, однако подобная простая схема выбрана сознательно для концентрации внимания на проблеме создания грид-приложений с использованием инструментария Alchemi.

Подготовка среды разработчика

Прежде всего, нужно подготовить среду разработки. Для этого необходимо выполнить следующие шаги:

- создать на машине разработчика минимальную “грид” из одного Менеджера и одного Исполнителя,

- проверить работоспособность “грид” с помощью одного или нескольких приложений-примеров,
- загрузить Alchemi SDK и распаковать в любую желаемую директорию,
- добавить путь к библиотеке `Alchemi.Core.dll` в переменную окружения `PATH`.

Создание класса грид-потока

Как обсуждалось ранее, вычислительную грид, построенную с помощью инструментария Alchemi, можно рассматривать как “виртуальную машину” с множеством процессоров. Программа (грид-приложение), предназначенная для такой “виртуальной машины”, представляет собой совокупность независимых параллельно выполняемых “частей” (грид-потоков).

Эти “части” параллельного исполнения (грид-потоки) являются экземплярами класса, унаследованного от `GThread`. Код, который должен быть выполнен в грид, определяется в методе `void Start()`.

Таким образом, чтобы создать новый грид-поток, необходимо объявить класс, наследуемый от класса `GThread`, и переопределить в нем метод `void Start()`. Перед объявлением класса обязательно нужно указать атрибут `Serializable`, который говорит о том, что экземпляр данного класса может быть отправлен по сети на другой компьютер:

```
[Serializable]
class CustromGridThread : GThread
{
    public override void Start()
    {

    }
}
```

Для решения поставленной задачи – определение простых чисел – нужно создать грид-поток, который будет проверять, является ли некоторое число простым.

Для этого необходимо проделать следующие шаги:

- создать новый проект консольного приложения, например, `PrimeNumberGenerator`,
- добавить ссылку на библиотеку `Alchemi.Core.dll` (ее следует указывать в любых проектах, использующих Alchemi API).

Теперь создадим класс грид-потока, который будет проверять число на делимость.

```
using System;
using Alchemi.Core;

namespace Tutorial
{
    /// <summary>
    /// Класс грид-потока, предназначенного для проверки
    /// целого числа на делимость.
    /// </summary>
    [Serializable]
    class PrimeNumberChecker : GThread
    {
        public int Candidate;
        public int Factors = 0;

        /// <summary>
        /// Создает новый класс грид-потока для проверки
        /// числа <code>candidate</code>.
        /// </summary>
        /// <param name="candidate"></param>
        public PrimeNumberChecker(int candidate)
        {
            Candidate = candidate;
        }

        /// <summary>
```

```
        /// Данный метод запускает поток на выполнение.
        /// </summary>
        public override void Start()
        {
            // Поиск всевозможных делителей числа
            for (int i = 1; i <= Candidate; i++)
            {
                if (0 == Candidate % i) Factors++;
            }
        }
    }
}
```

Создание класса грид-приложения

Для того, чтобы организовать параллельную работу независимых грид-потоков в грид-среде, воспользуемся классом `GApplication`.

```
class PrimeNumberGenerator
{
    // Объект GApplication является контейнером для грид-потоков
    // и предназначен для их запуска в грид
    public static GApplication App = new GApplication();

    [STAThread]
    static void Main(string[] args)
    {
        Random random = new Random();

        // Создание нескольких грид-потоков
        for (int i = 0; i < 100; i++)
        {
            App.Threads.Add(new PrimeNumberChecker(random.Next(1000000)));
        }

        // Инициаизация грид-приложения
        Init();

        // Запуск грид-приложения
        App.Start();

        Console.ReadLine();

        // Остановка грид-приложения
        App.Stop();
    }
}
```

Объект `GApplication` представляет собой контейнер для грид-потоков и необходим для взаимодействия с грид. В методе `void Main()` создадим 100 грид-потоков и добавим их к свойству `Threads` грид-приложения.

Приложение инициализируется с помощью метода `void Init()` (описан далее) и начинает выполняться, вызывая метод `void Start()` грид-потоков.

```
/// <summary>
/// Данный метод инициализирует грид-приложение.
/// </summary>
private static void Init()
{
    // Уставнока свойств соединения
    App.Connection = new GConnection("localhost", 9000, "user", "user");

    // Добавление зависимостей
```

```
    App.Manifest.Add(new
                     ModuleDependency(typeof(PrimeNumberChecker).Module));

    // Добавление обработчика события ThreadFinish
   App.ThreadFinish += new GThreadFinish(App_ThreadFinish);
}
```

Прежде, чем приложение может быть запущено (и потоки выполнены в грид), необходимо создать и установить подключение к грид-сети. Для подключения к грид-сети нужно знать:

- имя компьютера, на котором установлен Менеджер,
- порт Менеджера,
- имя пользователя,
- пароль.

Свойство `Manifest` объекта `GApplication` позволяет указать модули, необходимые для работы грид-потоков. Данные зависимости динамически загружаются Исполнителем во время первого выполнения потока. Зависимость, которую нужно обязательно добавить к свойству `Manifest` – это файл, содержащий реализацию грид-потока. Специальный класс `ModuleDependency` позволяет определять модули (`.dll` или `.exe` файлы) как зависимости.

После добавления зависимостей приложение готово к выполнению. Чтобы получать уведомления о завершении различных потоков, необходимо написать обработчик события `ThreadFinish`, который будет вызываться всякий раз, когда поток заканчивает своё выполнение.

```
/// <summary>
/// Данный метод вызывается при завершении потока.
/// </summary>
private static void App_ThreadFinish(GThread thread)
{
    // Приведение класса GThread к классу PrimeNumberChecker
    PrimeNumberChecker pnc = (PrimeNumberChecker)thread;

    // Проверяем, является ли число простым
    bool prime = false;

    if (pnc.Factors == 2)
        prime = true;

    // Отображаем результаты
    Console.WriteLine("Число {0} простое? {1} ({2} делителей)",
                      pnc.Candidate,
                      prime,
                      pnc.Factors);
}
```

Завершенный поток передается как параметр обработчику события `GThreadFinish`. Обработчик события приводит поток к типу `PrimeNumberChecker`, проверяет число делителей для определения, является ли число простым или нет, и, наконец, отображает результат.

Основные выводы

Близкие проекты

Инструментарий Alchemi, помимо своей реализации, основанной на сервисно-ориентированной архитектуре, имеет и другие особенности, отличающие его от других похожих проектов. В таблице 2 представлена сравнительная характеристика инструментария Alchemi с другими проектами, такими как Condor, SETI@Home, Entropia, GridMP и XtermWeb.

Таблица 1. Сравнительная характеристика инструментария Alchemi с другими проектами

	Alchemi	**Condor**	**SETI@home**	**Entropia**	**XtermWeb**	**Grid MP**
Архитектура	Иерархическая		Централизованная			
Возможность интеграции в глобальную грид	Да	Нет	Нет	Нет	Нет	Нет
Технологии реализации	C# , технология веб-сервисов	C	C++, Win32	C++, Win32	Java, Linux	C++, Win32
Поддержка мультикластера	Да	Да	Нет	Нет	Нет	Да
Глобальный брокер ресурсов грид	Да (Gridbus Broker)	Да (Condor-G)	Нет	Нет	Нет	Нет
Многопоточная модель программирования	Да	Нет	Нет	Нет	Нет	Нет
Интеграции приложений	Низкоуровневая (в основном)		Отсутствует (единственное применение)	Низкоуровневая (в основном)		

Система Condor была разработана в рамках проекта Condor Project (http://www.cs.wisc.edu/condor/) в университете Висконсина-Мэдисона. Данная система может применяться для управления кластером вычислительных узлов в выделенном режиме. Кроме того, специально разработанные механизмы позволяют системе Condor эффективно использовать свободные ресурсы простаивающих рабочих станций. Пользователи отсылают свои последовательные или параллельные задачи, после чего система помещает их в очередь и планирует выполнение на основе заданной политики, тщательно информируя пользователя о статусе задания и о его завершении. Поддерживаются операционные системы класса Windows и Unix.

SETI@Home (Search for Extra-Terrestrial Intelligence At Home) — научный некоммерческий проект распределённых вычислений, использующий свободные ресурсы на компьютерах добровольцев для поиска радиосигналов внеземного разума (http://setiathome.ssl.berkeley.edu/). Каждый пользователь персонального компьютера, подключённого к сети Интернет, может помочь проекту. Проект заключается в обработке данных радиотелескопа Аресибо на предмет поиска сигналов, которые можно интерпретировать как искусственные.

Инструментарий Entropia (http://www.entropia.com/) позволяет объединять обычные настольные компьютеры, работающие под операционной системы Microsoft Windows, в единый логический ресурс. Система имеет централизованное управление, при котором центральный менеджер задач управляет различными настольными клиентами. Специальный менеджер, устанавливаемый на отдельные узлы

сети, обеспечивает централизованный интерфейс для управления всеми клиентами Entropia-грид, доступ к которым можно получить с любого компьютера сети предприятия.

Проект XtermWeb (http://www.xtremwebch.net/) позволяет создать вычислительную сеть из множества равноправных ресурсов. В системе реализованы три основные сущности – Координатор, Исполнитель и Клиент. При помощи этих объектов конструируется так называемая XtermWeb-сеть. Клиент представляет собой специальную программу, позволяющую любому пользователю отсылать свои задачи в сеть. Задача, включающая в себя бинарные файлы и дополнительные файлы с параметрами, отсылается Координатору, после чего обрабатывается Исполнителями на свободно предоставляемых ресурсах.

Система Grid MP (http://www.ud.com/) позволяет организациям улучшить производительность своих ресурсоемких приложений, объединяя все доступные ресурсы предприятия. Архитектура системы в большей степени централизованная. Специальный сервис системы играет роль менеджера, принимая задачи от пользователей и планируя их выполнение на доступных ресурсах с предустановленными агентами. С помощью системы Grid MP можно объединить кластеры, рабочие станции и обычные настольные компьютеры.

Заключение

Инструментарий Alchemi, основанный на технологии Microsoft .NET, позволяет быстро и удобно сконструировать вычислительную грид-сеть и разработать соответствующие приложения. Благодаря поддержке выполнения грид-заданий на основе файлов имеется возможность интеграции в глобальную кросс-платформенную грид с использованием брокера ресурсов (системы распределения заданий в грид по конкретным узлам в соответствии с некоторыми критериями).

В будущем разработчиками панируется значительно расширить возможности инструментария в различных направлениях. Во-первых, планируется поддержка дополнительных функциональных возможностей API, включая взаимодействие между отдельными потоками. Во-вторых, ведутся работы над поддержкой мультикластера из равноправных Менеджеров. В-третьих, планируется поддержка различных политик выделения ресурсов с учетом экономичности, качества обслуживания и эксплуатационных соображений. В-четвертых, исследуется возможность перехода на стандарты OGSI посредством расширения существующего интерфейса управления заданиями. Этого, вероятно, удастся достичь с помощью интеграции с другими грид-системами, основанными на технологии Microsoft .NET (например, система OGSI.NET университета Виржинии) и соответствующим стандартам OGSI (Open Grid Services Infrastructure). Наконец, планируется реализовать возможности грида данных, что позволит использовать ресурсы данных в дополнение к вычислительным ресурсам.