УДК 681.324

М.А. Волк, К.В. Дема, В.В. Зозуля

Харьковский национальный университет радиоэлектроники, Харьков

АРХИТЕКТУРА МОДУЛЕЙ ГЕНЕРАЦИИ ПОТОКОВ ЗАДАЧ И ОЧЕРЕДИ В РАСПРЕДЕЛЕННОЙ СИСТЕМЕ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ GRID

В работе представлено описание принципов функционирования и внутренней организации компонентов генерации заявок и очереди задач для системы имитационного моделирования GRID инфраструктуры, приведена схема построения компонентов динамического учета статистической информации по работе очереди заданий.

Ключевые слова: *генерация потока заявок, конфигурационный файл, система массового обслуживания, очередь, статистический учет.*

Введение

В настоящие время наблюдается большой интерес у мирового научного сообщества к использованию интенсивно развивающихся GRID технологий. Подтверждением этому служит принятое в сентябре 2009 года постановление Кабинета Министров Украины об утверждении государственной целевой научно-технической программы "Внедрение и использование грид-технологий на 2009–2013 годы" [1]. Построение и использование больших вычислительных систем этого уровня требует большого внимания к эффективности их функционирования. Одним из наиболее используемых способов повышения эффективности процесса проектирования больших систем является имитационное моделирование. Однако, с учетом размерности решаемых в данном случае задач, организация имитационного моделирования является сложной задачей. Ряд подходов к организации имитационного моделирования в GRID приведено в [2]. В стадии разработки находится большое количество пакетов моделирования GRID-систем. Наиболее распространенными из них являются проекты Bricks [3], OptorSim [4] и GridSim [5]. Детальный сравнительный анализ этих пакетов приведен в [6]. Выводы, приведенные в последней работе, говорят об ограниченности данных систем моделирования. Основные их недостатки – проблемная ориентированность на решение частных задач (в основном, статистического характера) и специальная реализация алгоритмов на языках высокого уровня (конфигурационные файлы, Java). В Харьковском национальном университете радиоэлектроники ведется разработка системы имитационного моделирования GRID-систем GRASS (GRID Advanced Simulation System), которая позволит устранить приведенные выше недостатки. Назначение, структура и варианты использования этой системы в научных исследованиях подробно исследованы в [7]. Описание общей архитектуры системы, основанной на подключаемых модулях (плагинах – plug-in), приведено в [10].

В данной статье приводиться описание модулей системы GRASS, отвечающих за подключение моделей двух важных элементов GRID – источников и очереди задач, поступающих на обслуживание.

Генерация потока заявок

В распределенной имитационной модели для реализации функций генерации потока заявок в системе используются модули «Simple Tasks Manager» и «Simple Tasks Generator», при помощи которых можно генерировать поток заявок с динамически изменяющимися параметрами.

Модуль «Simple Tasks Generator» предназначен для генерации потока заявок с параметрами, изменяющимися по различным законам распределения. Данный модуль в обычном режиме работы генерирует заявки и добавляет их в очередь по мере генерации. Также имеется возможность записать результаты генерации в файл, чтобы впоследствии можно было повторить эксперимент с теми же данными. Модуль «Simple Tasks Manager» предназначен для генерации потока заявок на основе входного конфигурационного файла. Данный конфигурационный файл соответствует формату файла с данными эксперимента, который генерирует модуль «Simple Tasks Generator», что позволяет использовать описанные модули по-отдельности или совместно. В последнем случае возможно последовательное включение модулей в модели.

Модуль «Simple Tasks Manager»

Любой плагин [10] в системе должен поддерживать интерфейс Framework::IPlugin и реализовывать методы, заданные в данном интерфейсе. Вся логика генерации потока заданий расположена в

классе SimpleTasksManager, который включен в класс плагина. При старте системы плагин инициализируется и создает экземпляр класса SimpleTasksManager, передавая в конструктор параметры инициализации. Диаграмма классов модуля «Simple Tasks Manager» представлена на рис. 1.

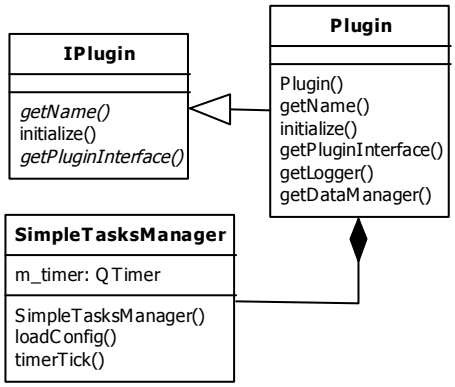

Рис. 1. Диаграмма классов модуля "Simple Tasks Manager

Класс Plugin является основным классом, описывающим модуль, подключаемый к системе распределенного имитационного моделирования GRID. Данный класс наследуется от интерфейса IPlugin и должен реализовать методы, описанные в этом интерфейсе. При запуске программы система подгружает требуемые плагины и инициализирует их, вызывая метод initialize() каждого экземпляра класса Plugin. Таким образом, все действия, которые необходимо выполнить на этапе загрузки модуля, необходимо описывать в данном методе. Метод getName() возвращает имя модуля, которое используется при регистрации всех плагинов в системе. Метод getPluginInterface() позволяет получить один из интерфейсов плагина по его имени. Также были описаны методы getLogger() и getDataManager(), которые позволяют быстро получить интерфейсы модулей Logger и DataManager соответственно. Они реализуют функции ведения журнала и работы с табличными структурами данных. Класс SimpleTasksManager определяет всю логику генерации потока заданий. На этапе инициализации плагина создается экземпляр класса SimpleTasksManager. При создании объекта класса SimpleTasksManager производится загрузка из файла конфигурации данных об эксперименте. Конфигурационный файл записан в формате XML.

При создании объекта класса SimpleTasksManager при помощи объекта QTimer планируется вызов метода timerTick(), в котором создается объект Task::Task очередного задания и этот объект отправляется в очередь (рис. 2). Объект задания создается при помощи фабрики [8] объектов Task::Factory, интерфейс которой предоставляет модуль очереди. Таким образом, данный модуль позволяет генерировать заявки с динамически изменяемыми параметрами, указанными в конфигурационном файле модуля. Это позволяет повторно проводить эксперименты с одними и теми же подготовленными данными.

Модуль «Simple Tasks Generator»

Данный модуль по структуре аналогичен модулю «Simple Tasks Manager». Вся логика генерации потока заданий расположена в классе SimpleTasksGenerator, который включен в класс плагина (рис. 3). При старте системы плагин инициализируется и создает экземпляр класса SimpleTasksGenerator, передавая в конструктор параметры инициализации.

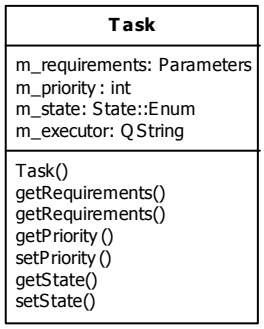

Рис. 2. Класс Task::Task

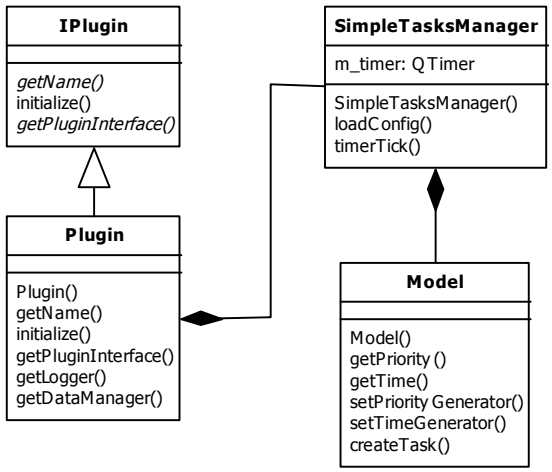

Рис. 3. Диаграмма классов модуля «Simple Tasks Generator»

При инициализации класса SimpleTasksGenerator вызывается метод loadModel(), в котором данные о модели эксперимента читаются из конфигурационного файла и на их основе инициализируются параметры модели (устанавливаются параметры объекта m_model – экземпляр класса Model). Также как и в модуле «Simple Tasks Manager», в данном модуле используется объект QTimer для генерации потока заявок с различными задержками во време-

ни. Метод timerTick() используется для создания объекта Task::Task очередного задания и этот объект отправляется в очередь (если модуль работает в обычном режиме). При включенном режиме генерации файла эксперимента в данном методе также производится запись данных в файл в специальном формате. Модель хранит в себе информацию о законах распределения и позволяет генерировать параметры заявок в соответствии с этими законами. В базовую версию включены следующие законы распределения: нормальный, экспоненциальный, равномерный, константный и их комбинации. Кроме того, разработчик обладает возможностью подключения своего плагина, реализующий новый оригинальный закон распределения.

Модуль Queue

Моделирование процессов кластерной, многоузловой компьютерной обработки и распределения потоков задач глобально представляет собой классическую систему массового обслуживания (СМО) [9]. В системе моделирования GRID, модуль Queue соответствует элементу «очередь» структуры СМО. Разные алгоритмы функционирования очереди позволяют получить различные результаты эффективности работы того или иного алгоритма распределения задач и ресурсов. На данный момент модуль Queue реализует простейшую схему работы очереди с неограниченным числом свободных мест. Основной интерфейс взаимодействия с модулем предоставляет семь основных методов:

- *removeTask(), pushTask* – добавляют и удаляют задание из очереди;
- *browse()* – позволяет получить доступ ко всем заданиям в очереди на текущий момент;
- *browseStatistic()* – позволяет получить все элементы статистики. С помощью метода *getValue()* можно получить текущее значение того или иного элемента;
- *getTaskCount()* – возвращает количество заданий в очереди на данный момент;
- *addEventsListener/removeEventsListener* – позволяет любому другому модулю отслеживать вызовы методов добавления или удаления заданий очереди для любых действий, связанных с реагированием системы на внутренние изменения.

Модуль представлен основным классом, реализующим непосредственные функции очереди и функции инициализации статистических элементов. С помощью класса Plugin модуль подключается к глобальной среде выполнения системы моделирования. Структура данных компонентов представлена на рис. 4. IQueue – класс-интерфейс, подключая который, осуществляется взаимодействие с очередью. Класс-реализация Queue содержит коллекцию, в которой хранятся поступающие на обработку задания. Получение списка заданий из коллекции реализуется с помощью итератора, который возвращает функция *browse()*. Аналогичный механизм реализован для хранения и доступа к объектам элементов статистики. Сбор статистической информации осуществляется несколькими классами, каждый из которых осуществляет вычисление своего конкретного параметра статистики: интенсивность входного и выходного потоков, загруженность очереди, время работы, количество обработанных и необработанных заданий и другую информацию, которая накапливается во время работы. Структура классов статистики представлена на рис. 5.

Каждый элемент статистики содержит указатель на класс Counter, выполняющий функции подсчета вызовов *removeTask(), pushTask* очереди, времени работы системы. На основе этих данных происходит вычисление нужных значений. Доступ к значениям осуществляется с помощью виртуального метода *getValue()*. Пользователь не имеет доступа непосредственно к объектам элементов, а взаимодействует с системой через интерфейс IStatistic. Реализация самого объекта задания выполнена в виде структуры данных, содержащих в качестве полей информацию об основных ключевых параметрах: приоритет задания и перечень ресурсов, востребованных для выполнения. Также задание имеет поле статуса выполнения, характеризующее её состояние в текущий момент. Перечень ресурсов представляет собой динамическую структуру, абстрактно представляющую собой таблицу вида «имя – значение», неограниченную по длине и типу данных. Таким образом, имеется возможность гибкого определения параметров задач для различных экспериментов. Структурная схема классов задания показана на рис. 6. Класс IFactory предоставляет интерфейс для создания заданий. Данных механизм является базовым и будет расширяться для успешного моделирования более сложных процессов, с комплексными начальными условиями и характеристиками заданий.

Выводы

Созданное программное обеспечение дает возможность проводить научные исследования по моделированию элементов GRID-систем. Для исследователей в области систем массового обслуживания возможно проведение экспериментов по анализу математических моделей потоков заданий, управлению очередями, определению нагрузочных характеристик отдельных модулей, моделей предоставления трафика и т.п. При проектировании конкретных сегментов GRID можно промоделировать их работу в зависимости от разных потоков заявок, стратегий распределения задач и различных конфигураций ресурсов. Гибкость разработанной архитектуры позволяет расширять или модифицировать описанные

модули с целью проведения более сложных экспе- риментов.

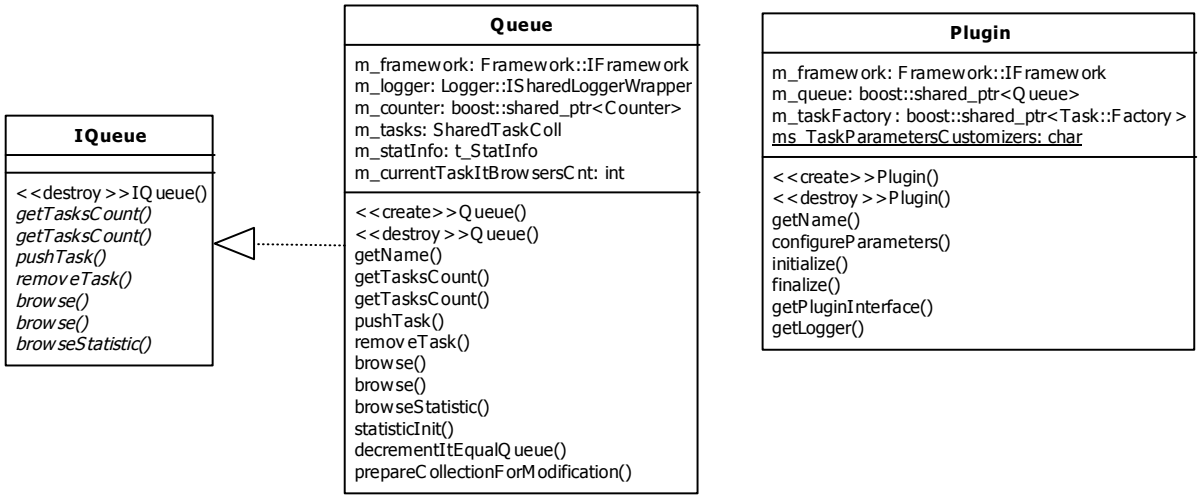

Рис. 4. Структура модуля Queue

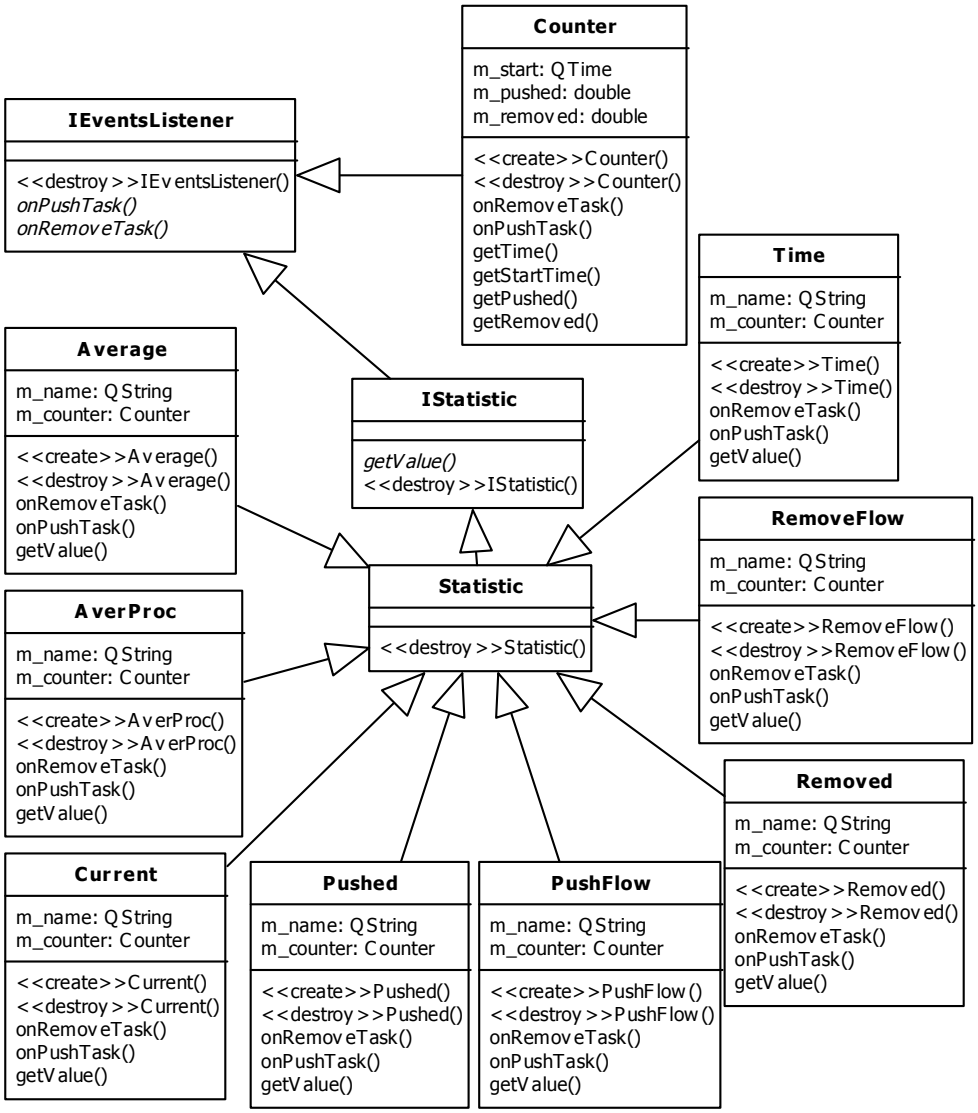

Рис. 5. Структура классов сбора статистической информации

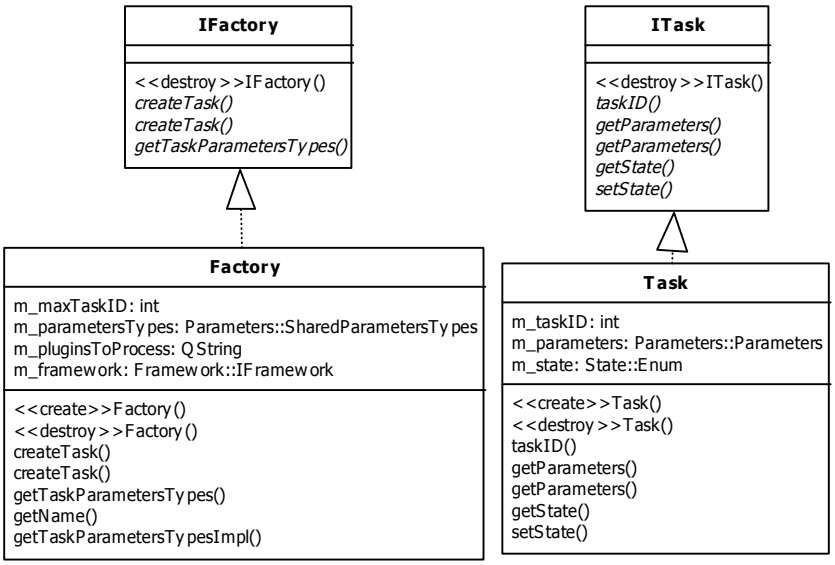

Рис. 6. Структурная схема классов задания

Список литературы

1. Про затвердження Державної цільової науково-технічної програми впровадження і застосування грід-технологій на 2009-2013 роки / Постанова Кабінету Міністрів України, №1020, 23.09.2009.

2.Волк М.А. Структурная организация поведенческого имитационного моделирования в grid // Системи обробки інформації. – Х.: ХУ ПС, 2007. – Вип. 9(67). – С. 41-45.

3. Bricks: A Performance Evaluation System for Grid Computing Scheduling Algorithms. [Электронный ресурс]. – Режим доступа к статье: http://ninf.apgrid.org.

4. Simulating data access optimization algorithms // OptorSim. [Электронный ресурс]. – Режим доступа к статье: http://edg-wp2.web.cern.ch.

5. GridSim: A Grid Simulation Toolkit for Resource Modelling and Application Scheduling for Parallel and Distributed Computing. [Электронный ресурс]. – Режим доступа к статье: http://www.gridbus.org.

6. Коренков В.В. Пакеты моделирования DataGrid [Электронный ресурс] / В.В. Коренков, А.В. Нечаевский // Системный анализ в науке и образовании. – 2009. – №1. – Режим доступа к статье: http://sanse.ru.

7. Волк М.А. Структура программного комплекса имитационного моделирования элементов GRID-систем для научных исследований // Системи обробки інформації. – Х.: ХУ ПС, 2009. – Вип. 3(77). – С. 125-128.

8. Троелсен Эндрю. C# и платформа .NET. Библиотека программиста / Эндрю Троелсен. – СПб.: Питер, 2004. – 796 с.

9. Hamdy A. Taha. Operations Research: An Introduction, Prentice Hall; 8th. Edition. – 2006. – 813 p.

10. Волк М.А. Архитектура имитационной модели GRID – системы, основанная на подключаемых модулях / М.А. Волк, А.С. Горенков // Проблеми інформатики і моделювання. Матеріали дев'ятої міжнародної науково-технічної конференції. – Х.: НТУ "ХПІ", 2009. – 92 с.

Поступила в редколлегию 27.01.2010

Рецензент: д-р техн. наук, проф. С.Г. Удовенко, Харьковский национальный университет радиоэлектроники, Харьков.

АРХІТЕКТУРА МОДУЛІВ ГЕНЕРАЦІЇ ПОТОКІВ ЗАВДАНЬ І ЧЕРГИ В РОЗПОДІЛЕНІЙ СИСТЕМІ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ GRID

М.О. Волк, К.В. Дема, В.В. Зозуля

У роботі представлено опис принципів функціонування та внутрішньої організації компонентів генерації заявок і черги завдань для системи імітаційного моделювання GRID інфраструктури, надана схема побудови компонентів динамічного обліку статистичної інформації за роботою черги завдань.

Ключові слова: *генерація потоку заявок, конфігураційний файл, система масового обслуговування, черга, статистичний облік.*

ARCHITECTURE OF TASK GENERATION MODULE AND QUEUE MODULE IN DISTRIBUTED IMITATING MODEL OF GRID SYSTEM

M.A. Volk, K.V. Dema, V.V. Zozulya

Description of task generator and queue inner structure and functionality for GRID simulation system is presented. Scheme of dynamic statistic accounting component in queue module is introduced.

Keywords: *generation of stream of requests, configuration file, queuing system, turn, statistical account.*