ИНФОРМАТИКА И ПРОГРАММИРОВАНИЕ

УДК 004.415.2

К.Ю. Войтиков, П.Н. Тумаев

ОСНОВНЫЕ ЭЛЕМЕНТЫ АРХИТЕКТУРЫ КОМПОНЕНТА «КЛИЕНТ» ДЛЯ СИСТЕМЫ РАСПРЕДЕЛЕННЫХ РАСЧЕТОВ

Работа посвящена проектированию клиентского приложения для работы с сервером распределенных расчетов. Рассматриваются вопросы внутреннего устройства компонента, организация работы приложения по расписанию, реализация механизма модульного расширения системы, а также интерфейс веб-службы сервера, с которым будет работать данное приложение.

Ключевые слова: *распределенные вычисления*, *GRID*, *WCF*, *имитационное моделирование*, *объектно-ориентированное проектирование.*

Рост вычислительной мощности современных компьютеров – не только непрерывный, но и постоянно ускоряющийся процесс. Растущие возможности суперкомпьютеров каждый год позволяют решать все более сложные задачи. Но помимо промышленных суперкомпьютеров не менее быстрыми темпами развивается сектор ПК и компьютерных сетей общего назначения. И существует целый пласт задач, также нуждающихся в огромных вычислительных мощностях, но практически решаемых при помощи задействования свободного времени обычных ПК.

В [1] была представлена компонентная модель объектно-ориентированной системы имитационного моделирования. Ключевыми компонентами системы являются компоненты «Сервер» и «Клиент». Компонент «Сервер» отвечает за координацию работы компонентов «Клиент», получение заданий от пользователей системы, разделение заданий на подзадания для конкретных «Клиентов» и итоговую обработку результатов всех подзаданий для получения конечного результата, необходимого пользователю.

1. Компонент «Сервер»

Для достижения высокой степени повторной используемости компонентов системы, а точнее, для реализации возможности использовать однажды развернутую систему для выполнения любых расчетов, реализован механизм расширения вычислительной функциональности системы при помощи подключаемых модулей [2].

В состав компонента «Сервер» входят следующие компоненты:

- Task WCF service – Windows Communication Foundation (WCF) web-служба сервера для получения заданий и выдачи результата пользовательским приложениям;
- GRID-worker WCF service – WCF web-служба для подключения компонентов «Клиент»;

• GRID-server Logic – «ядро» компонента, в котором описаны классы предметной области системы и сценарии манипуляции ими;

• Dividers – набор алгоритмов разделения заданий на подзадания. Предусматривает добавление новых алгоритмов;

• Storage – поставщик хранилища для хранения заданий, подзаданий, результатов и учетной информации о существующих «Клиентах». Потенциально возможно создание поставщиков хранилища для любых систем хранения информации – баз данных, бинарных или XML-файлов, облачных хранилищ и т.д.;

• Modules – каркас модульного расширения системы и набор самих модулей.

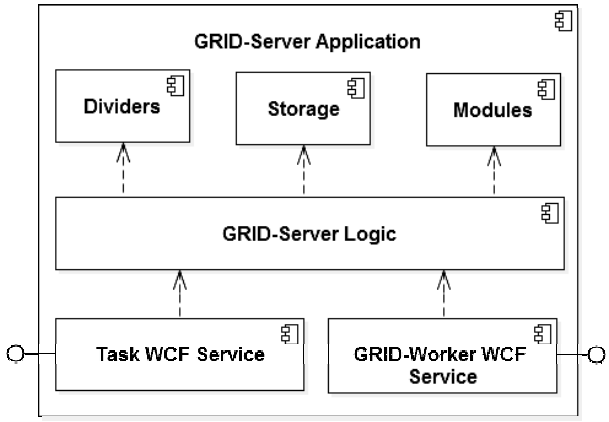

Рис. 1. Структура компонента «Сервер»

Важно отметить, что компонент «Сервер» в данном случае является пассивным. Он предоставляет два интерфейса – для клиентов системы и для компонентов «Клиент», при этом сам не производит никаких обращений к другим компонентам системы. Такой подход позволяет сделать сервер максимально доступным для большого количества компьютеров, находящихся в разных условиях сетевого доступа к физическому серверу.

2. Элементы интерфейса веб-службы сервера распределенных расчетов

В свою очередь за сам процесс выполнения расчетов и отправку результатов на сервер отвечает компонент «Клиент».

Рассмотрим интерфейс веб-службы [3] сервера распределенных расчетов, с которым придется работать данному клиенту. Очевидно, что набор сервисов, предоставляемых службой, напрямую зависит от принципов взаимодействия с ее помощью клиента и сервера. Так как принципиально сервер расчетов является пассивным хранилищем заданий и результатов, основными действиями клиента по отношению к нему будут запрос и получение доступных заданий, а также отправка результатов их выполнения. Однако набор необходимых действий этим не ограничивается. Прежде всего, каждый новый клиент должен быть зарегистрирован в системе. Сервер расчетов хранит коллекцию данных обо всех зарегистрированных клиентах, их производительности и скорости сетевого соединения, ведет статистику их работы и на основе этих данных составляет наиболее оптимальные для

каждого клиента списки заданий. Из этого также следует, что каждый клиент должен как минимум однажды провести тест производительности компьютера, на котором он установлен и скорость сетевого соединения с сервером, после чего передать эти данные серверу. В дальнейшем при необходимости возможно повторное проведение таких тестов и обновление сведений на сервере. Таким образом, основными методами службы сервера являются:

- RegisterClient() – регистрация клиента;
- SendRatings() – отправка данных о производительности;
- GetGridTasks() – получение заданий;
- SendGridResults() – отправка результатов.

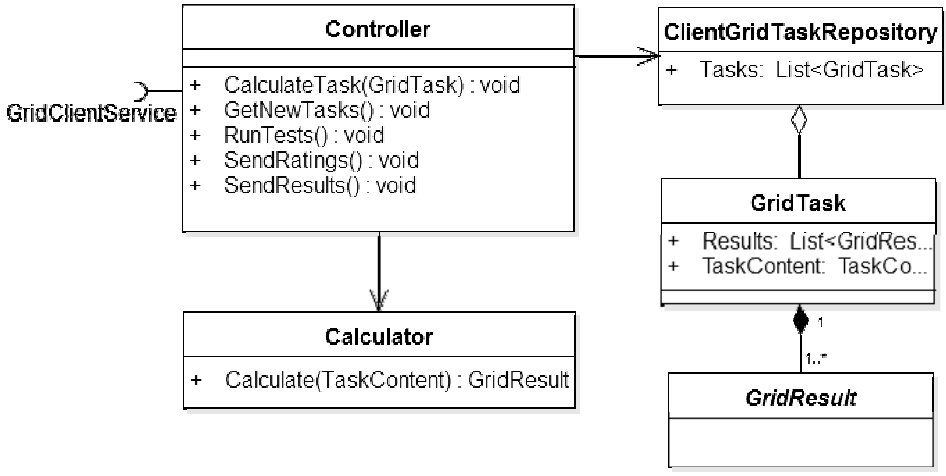

Рис 2. Структура компонента «Клиент»

Кроме этого, служба предоставляет еще несколько методов вспомогательного назначения для проверки, нуждаются ли какие-то результаты в повторной отправке, и для получения журнала выполненных клиентом заданий и «списка ожидания» сервера. Эти методы существуют для обеспечения отказоустойчивости системы в непредвиденных случаях.

Рассмотрим внутреннее устройство компонента «Клиент», представленного на рис. 2. Подобно компоненту «Сервер» задания и результаты их выполнения хранятся в виде экземпляров классов GridTask и GridResult соответственно. Подобно GridTaskRepository на сервере, здесь интерфейсом внешнего взаимодействия с данными коллекциями служит экземпляр класса ClientGridTaskRepository, немного отличающийся от первого набором предоставляемых функций.

3. Организация работы компонента «Клиент»

Для координации работы всего компонента предназначен класс Controller. Экземпляр класса непосредственно совершает вызовы методов веб-службы сервера, диспетчеризирует получение новых заданий и отправку результатов, инициирует выполнение заданий и запуск тестов.

В организации сеансов связи с сервером важную роль играет то, что к работе системы планируется привлечь так называемый «volunteer computing» – предос-

тавление сторонними пользователями свободного времени личных ПК для нужд распределенных расчетов. В этом случае каждый пользователь, и только он, должен иметь контроль над временем и условиями как совершения самих расчетов, так и передачи данных по сети. Для этой цели служит класс Scheduler (рис. 3), хранящий в виде коллекции объектов :Schedule заданные пользователем временные расписания работы или другие условия ее автоматического запуска/остановки (например, определенное время простоя ЦП). Объект Controller, в свою очередь, с заданным промежутком времени выполняет проверку, существует ли условие, согласно которому, в данный момент нужно совершить то или иное действие.

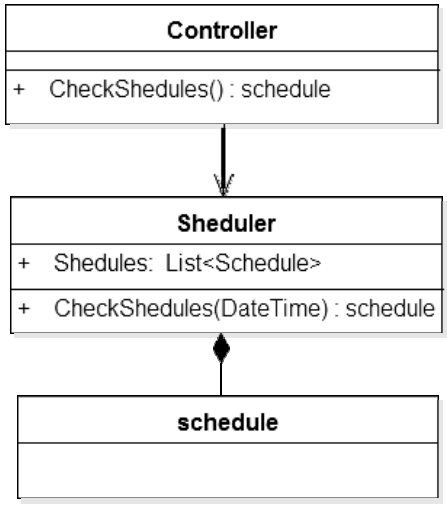

Рис 3. Реализация расписания работы компонента

Представленный в [1] механизм модульного расширения системы на стороне клиента реализуется следующим образом. Объекты GridResult, Calculator, а также TaskContent, инкапсулированные в GridTask, не являются прямыми экземплярами этих абстрактных классов, вместо этого объекты инстанцируют конкретные классы-потомки, описанные в подключаемом модуле. Так, TaskContent имеет структуру и набор свойств, характерных для конкретного класса задач, GridResult представляет собой объект для хранения результатов, специфичных для такой задачи, а Calculator является непосредственным исполнителем необходимых для получения результата расчетов.

Заключение

В статье разработана архитектура клиентского компонента системы распределенных расчетов, отвечающая всем требованиям, характерным для такого рода задач. Предоставляемая пользователю возможность лично контролировать процесс использования времени компьютера призвана привлечь к работе системы большое количество клиентов. Применение для связи компонентов системы технологии Microsoft WCF [3] облегчает соблюдение требований к конфигурации сети на стороне клиента, что значительно увеличивает количество ПК, потенциально доступных для использования.

ЛИТЕРАТУРА

1. *Войтиков К.Ю.*, *Моисеев А.Н.*, *Тумаев П.Н.* Компонентная модель распределенной объектно-ориентированной системы имитационного моделирования // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2010. № 1(10). С. 78–83.
2. *Войтиков К.Ю.*, *Тумаев П.Н.* Построение архитектуры сервера распределенных вычислений // Научное творчество молодёжи: материалы XIV Всероссийской научно-практической конференции (15 – 16 апреля 2010 г.). Томск: Изд-во Том. ун-та, 2010. Ч. 1. С. 115–118
3. *Резник С.*, *Крейн Р.*, *Боуэн К.* Основы Windows Communication Foundation для .NET Framework 3.5: пер с англ. М.: ДМК Пресс, 2008. 480 с.

Войтикоков Константин Юрьевич
Тумаев Павел Николаевич
Филиал Кемеровского государственного университета
в г. Анжеро-Судженске.
E-mail: kost_v@ngs.ru; ptumaev@yandex.ru

Поступила в редакцию 19 ноября 2010 г.