Особенности взаимодействия элементов архитектуры расширяемой системы распределенных вычислений

Работа выполнена в рамках аналитической ведомственной целевой программы «Развитие научного потенциала высшей школы (2009-2010 годы)», проект № 4761

К.Ю. Войтиков
Филиал Кемеровского государственного университета в г. Анжеро-Судженске,
kost_v@ngs.ru

П.Н. Тумаев
Филиал Кемеровского государственного университета в г. Анжеро-Судженске,
Pavel.Tumaev@gmail.com

Аннотация

В докладе рассматривается взаимодействие элементов объектно-ориентированной Desktop-GRID системы при выполнении основных сценариев ее использования, таких как: добавление заданий для системы, разделение задания на подзадания для выполнения в распределенной среде, получение и выполнение подзаданий, а также отправка результатов на сервер компонентами распределенной среды, итоговая обработка массива промежуточных результатов сервером и выдача конечного результата.

ODIS Drops – объектно-ориентированная система распределенных вычислений, предоставляющая необходимый каркас расширения для решения любых задач, отвечающих некоторым требованиям разделяемости и описания [1]. В том числе для работы в рамках системы распределенных вычислений могут быть адаптированы существующие инструменты для решения конкретных задач: отдельные алгоритмы и классы на десятках .NET-совместимых языков, .NET- и COM-библиотеки, любые другие интероперабельные приложения.

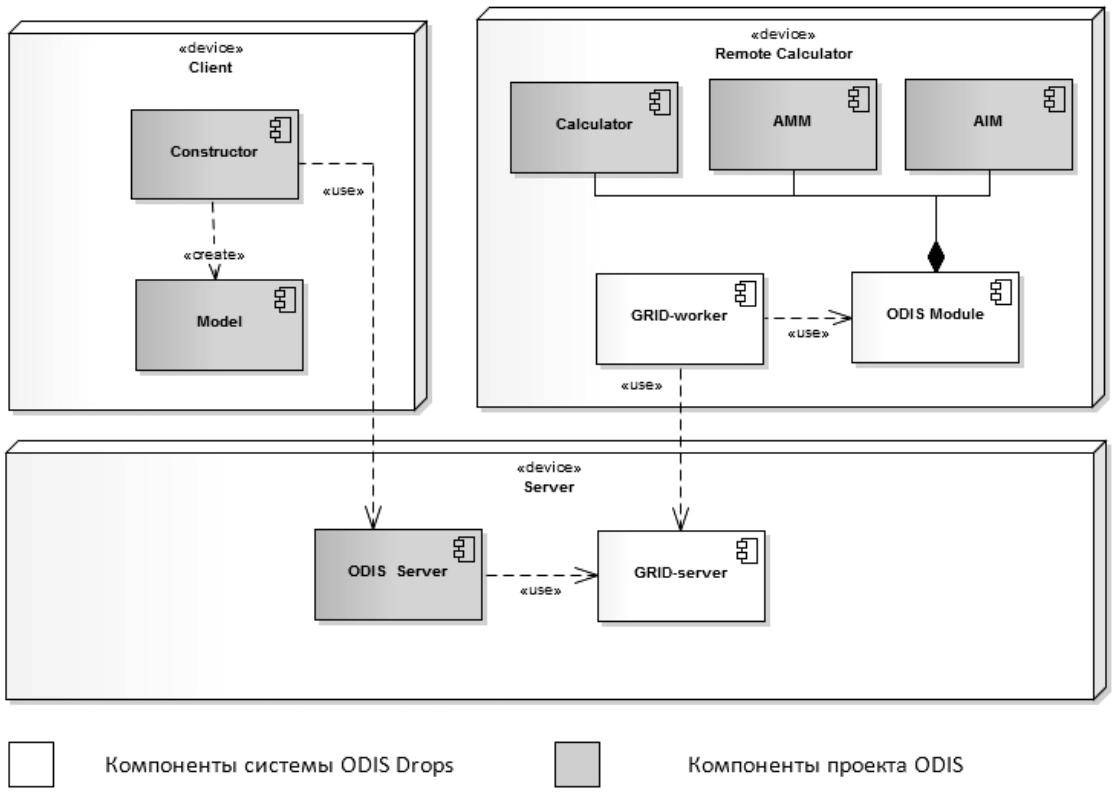

Рис. 1 Компоненты системы.

На рис. 1 показано распределение компонентов системы по трем узлам: Client – компьютер пользователя системы, составляющего задания и получающего конечные результаты их выполнения; Remote Calculator – компьютер, ресурсы которого используются для осуществления расчетов; Server – компьютер, посредством которого осуществляется координация остальных узлов.

Ключевыми компонентами системы являются GRID-server и GRID-worker. Компонент GRID-worker устанавливается на компьютеры, используемые в качестве ресурса, и отвечает непосредственно за сам процесс расчетов, получение новых заданий и отправку результатов на сервер. Компонент GRID-server устанавливается на сервер сети распределенных вычислений и отвечает за координацию работы объектов GRID-worker, получение заданий от пользователей системы, разделение заданий на подзадания для конкретных объектов GRID-worker и итоговую обработку результатов всех подзаданий для получения конечного результата, необходимого пользователю.

Важно отметить, что Grid-server в данном случае является пассивным. Он предоставляет два интерфейса – для клиентов системы и для объектов GRID-worker, при этом сам не производит никаких обращений к другим компонентам системы. Вместе с технологией Microsoft WCF, выбранной для обеспечения связи между компонентами это

позволило сделать сервер максимально доступным для большого количества компьютеров, находящихся в разных условиях сетевого доступа к физическому серверу.

Для решения различных типов задач на однократно развернутой сети объектов GRID-worker используется механизм модульного расширения системы [2].

Поведение объектов системы в процессе работы представлено на рис. 2,3,4.

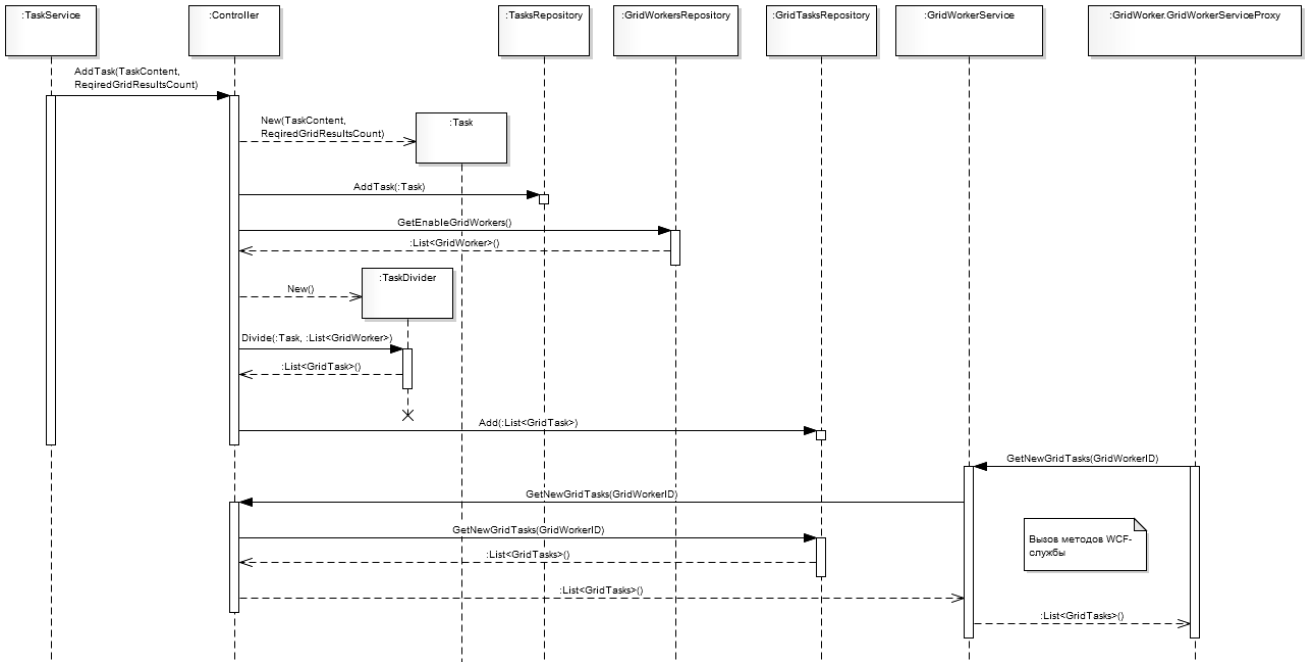

Рис. 2 Поведение объектов системы при добавлении новой задачи

На рис. 2 отражено поведение объектов, входящих в состав компонента GRID-server, в процессе добавления клиентским приложением нового задания, разделения задния на подзадания и отправку поздаданий для решения компонентами GRID-worker. Как видно из рисунка, инициатором создания нового задания в системе является объект TaskService, который, получив соответствующее сообщение от внешнего клиентского ПО, передает объекту Controller экземпляр класса TaskContent. Фактически же в этот момент передается экземпляр класса-потомка от TaskContent, описанного в одном из модулей расширения и описывающего структуру конкретного типа задач.

Вместе с содержимым задания объект web-службы передает параметр ReqiredGridResultsCount, указывающий, какое количество промежуточных результатов необходимо накопить перед итоговой обработкой и получением общего решения. Далее компонет Controller создает новый экземпляр класса Task, помещая в него полученные от web-службы данные и добавляет его к коллекции всех заданий системы – объекту TaskRepository. Для разделения нового задания на подзадания для распределенных компонентов, объект Controller запрашивает данные о существующих и доступных для работы объектах GRID-worker из объекта-хранилища GridWorkersRepository в виде объектов GridWorkerInfo. После этого объект Controller создает экземпляр одного из установленных в системе делителей заданий – потомков класса TaskDivider и передает ему требующее разделения задание и список данных об объектах GRID-worker, в результате чего ему возвращается список объектов подзаданий GridTask. Каждый объект GridTask во первых содержит в себе копию объекта TaskContent, описывающего задание, а во вторых в любой

момент времени после своего создания содержит ссылку на один из объектов GridWorkerInfo и параметр ReqiredGridResultsCount аналогичный параметру, полученному от web-службы. Таким образом любой объект-поздазание GridTask всегдя является «порученым» конкретному исполнителю Grid-worker с указанием количества результатов, ожидаемых от этого исполнителя. Очевидно, что при этом суммарное количество всех промежуточных результатов ожидаемых от всех задействованых в выполнении задания исполнителей равняется необходимому для итоговой обработки всего задания общему количеству промежуточных результатов. После разделения все полученные подзадания помещаются в хранилище подзаданий – объект GridTasksRepository, и становятся доступными для передачи исполнителям. После этого компонент Grid-server находится в состоянии ожидания обращений компонентов Grid-worker, установленных на компьютерах-ресурсах. Такие обращения происходят посредством второй web-службы сервера GridWorkerService.

Процессы происходящие с подзаданием на стороне компонента Grid-worker представлены на рис.3

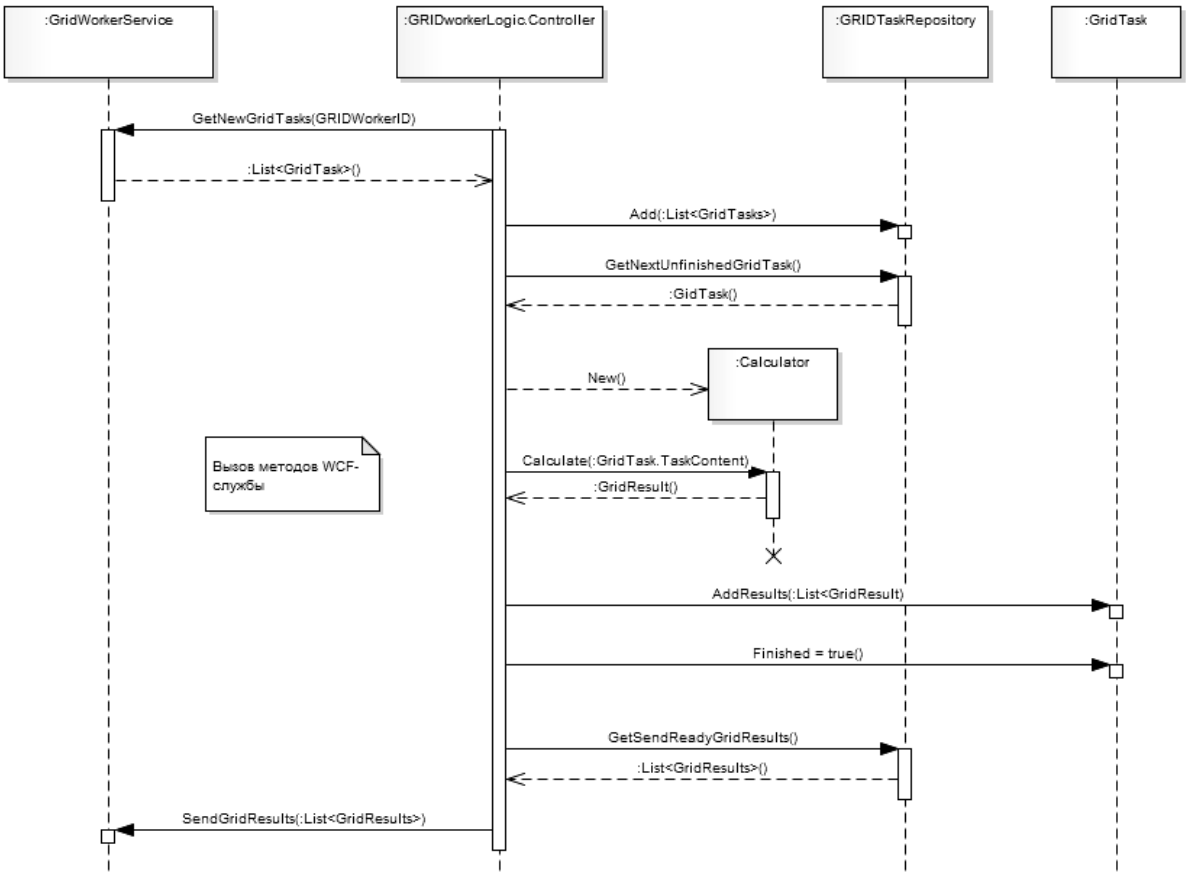

Рис. 3 Поведение объектов системы при выполнении расчетов

Как было сказано выше, получение новых заданий происходит путем обращения к web-службе сервера GridWorkerService. При запросе новых заданий объект Controller (не путать с объектом Controller в составе компонента Grid-server) передает службе свой уникальный идентификатор. На стороне сервера это ведет к поиску всех незавершенных заданий, порученных данному исполнителю Grid-worker, которые еще не были ему отправлены, после чего список объектов GridTask, удовлетворяющих таким условиям, возвращается объекту Controller. Получив список объектов-заданий объект Controller

помещает их в локальное хранилище всех порученных данному исполнителю подзаданий – объект GridTaskRepository.

Далее при наступлении условий заданных пользователем компьютера, на котором установлен компонент Grid-worker (например необходимое время простоя компьютера или определенное время суток) объект Controller инициирует выполнение подзаданий. Для этого из хранилища подзаданий запрашивается следующее невыполненное подзадание вызовом метода GetNextUnfinishedGridTask(). Если в хранилище имеются невыполненные подзадания, одно из них возвращается объекту Controller. Для решения подзадания объект Controller создает экземпляр одного из классов-потомков Calculator, описанного в одном из модулей расширения и описывающего процесс выполнения данного конкретного типа заданий. Как было сказано выше, вся информация, необходимая для выполнения подзадания инкапсулируется в нем в виде объекта TaskContent. Именно этот объект передается в только что созданный объект Calculator для совершения расчетов. В ответ объект Controller получает экземпляр класса-потомка GridResult, также описанного в модуле и описывающего сущность промежуточного результата для данного типа задач. Далее объект-результат добавляется к списку результатов объекта GridTask, для которого производились расчеты и в случае если количество этих результатов в списке удовлетворяет условию ReqioredGridResultCount, данное задание помечается как завершенное.

Так же во время наступления определенных пользователем компьютера-ресурса условий (в данном случае это может быть наличие простаивающего интернет-подключения или так же определенное время суток) производится отправка накопленных результатов на сервер. Для этого объект Controller запрашивает из хранилища подзаданий готовые к отправке результаты, передает их серверу посредством web-службы и помечает данные результаты в хранилище как отправленные. Хранить ли результаты после отправки на сервер, в течение какого времени или при каких других условиях (например объем свободного дискового пространства) также решает пользователь компьютера-ресурса, устанавливая соответствующие настройки компонента Grid-worker.

Накопление и последующая обработка промежуточных результатов сервером представлена на рис. 4

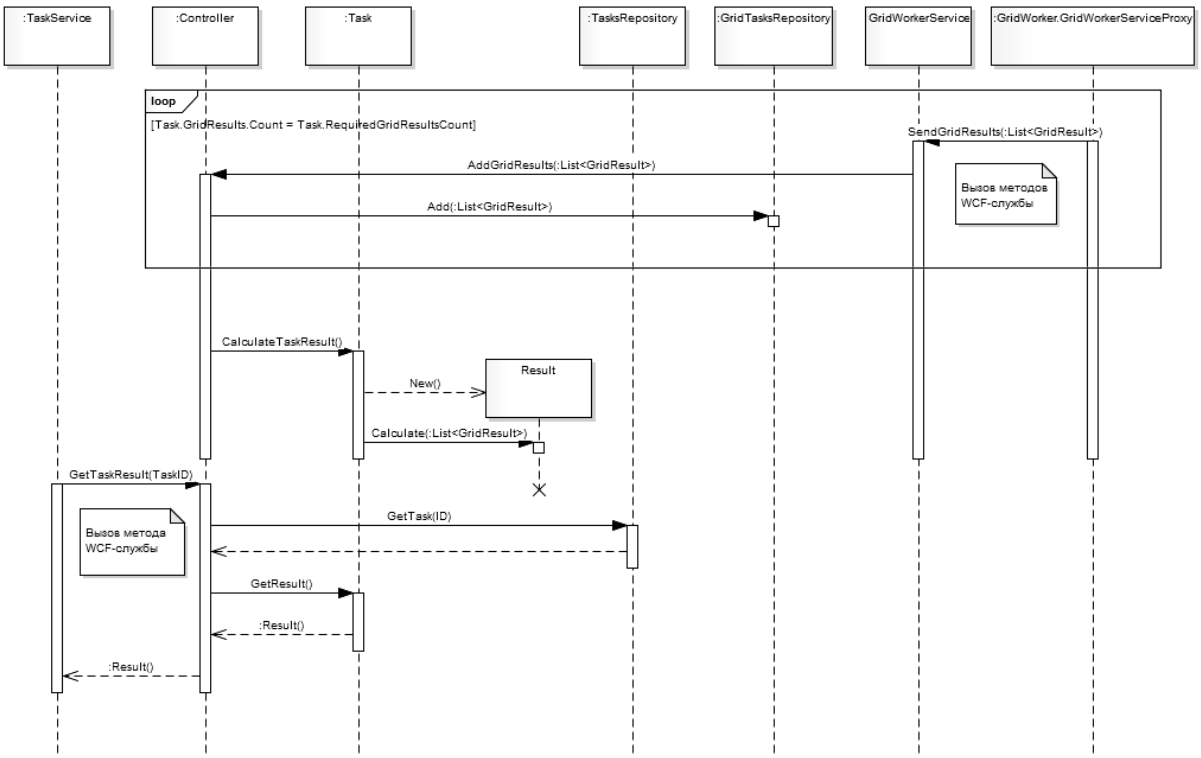

Рис. 4 Поведение объектов системы при итоговой обработке результатов

Получив от web-службы GridWorkerService список результатов подзаданий, объект Controller сервера распределяет их, добавляя к списку результатов соответствующих объектов GridTask в хранилище подзаданий GridTaskRepository. Так продолжается до тех пор, пока суммарное количество результатов подзаданий определенного задания не станет достаточным для его итоговой обработки. После чего объект Controller инициирует итоговую обработку задания. Для этого объектом Task, для которого выполняется обработка, вновь создается экземпляр одного из классов-расширений, описанных в подключенном модуле. На этот раз таким классом является класс-потомок Result, описывающий как структуру итогового результата выполнения задания в целом так и сценарий его получения из массива промежуточных результатов. После создания объекта Result объект Task передает ему собственный список накопленных промежуточных результатов и инициирует их обработку. В результате объект Result наполняется значениями свойств, описанных в модуле расширеня, и непосредственно являющихся результатом выполнения задания, необходимым его автору.

После этого серверу остается лишь дождаться обращения клиентского приложения к службе TaskService за результатами конкретного задания. После чего найти это задание в хранилище TaskRepository и вернуть инкапсулированный в него объект Result.

Представленные модели поведения системы наглядно демонстрируют практическое использование механизма ее расширения. Благодаря такому механизму рассмотренные сценарии работы системы являются едиными и не должны будут изменяться в будущем для решения любых задач целевого подмножества.

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, 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.