Реализация проблемно-ориентированных вычислительных сервисов в среде MathCloud*

О.В. Сухорослов

Центр грид-технологий и распределенных вычислений ИСА РАН

В статье рассматриваются принципы реализации и основные компоненты сервис-ориентированной научной среды MathCloud. Целями данной среды являются предоставление унифицированного доступа к проблемно-ориентированным вычислительным сервисам и поддержка интеграции данных сервисов при решении прикладных задач. Во главу предлагаемого подхода к реализации среды MathCloud ставятся удобство разработки сервисов, простота доступа к сервисам пользователей и использование открытых технологий.

1. Введение

Современные грид-системы ориентированы на интеграцию высокопроизводительных вычислительных ресурсов для решения задач с предельно высокими требованиями к подобным ресурсам, а также задач, допускающих декомпозицию на множество небольших независимых подзадач. Представляется, что концепция грид-вычислений может быть использована для решения более широкого класса задач, отличительной особенностью которых является возможность их декомпозиции на относительно «крупные» типовые подзадачи. Данный класс фактически охватывает широкий спектр вычислительных задач математики, физики, химии, биологии и т.д. Для решения таких задач требуется набор сервисов решения типовых вычислительных математических задач, связанных друг с другом в соответствии со схемой решения исходной задачи. Появление возможности решения сложных задач путем композиции распределенных проблемно-ориентированных сервисов способно вывести распределенные вычислительные среды на качественно новый уровень.

Кроме того, все большее распространение получает модель научной кооперации и разделения труда, основанная на совместной работе территориально распределенных, часто - международных, коллективов исследователей. В подобных «распределенных» научных проектах остро стоит проблема совместного использования наработок сторон, вовлеченных в проект. Чаще всего речь идет о тех или иных вычислительных пакетах, приложениях и моделях, а также архивах и базах данных. Эта проблема также может быть решена путем преобразования данных наработок в удаленно доступные проблемно-ориентированные сервисы. Другими ситуациями, в которых может возникать потребность в создании подобных сервисов, являются публикация и обмен результатами научных исследований, внедрение и коммерциализация полученных результатов и создание образовательных ресурсов.

Предлагаемый подход, охватывающий фактически все этапы научных исследований, состоит в построении распределенных вычислительных сред нового поколения, предоставляющих доступ к проблемно-ориентированным сервисам и образующих универсальную инфраструктуру для научной кооперации. Данная инфраструктура базируется на сервис-ориентированном подходе: пользователи преобразуют свои приложения в удаленно доступные сервисы, которые могут быть обнаружены и использованы другими пользователями для решения интересующих их задач.

Актуальность данного направления исследований подтверждается сформулированной в 2005 году концепцией «сервис-ориентированной науки» (Service-Oriented Science) [1], автором которой выступил Иэн Фостер, один из основоположников грид-вычислений. В соответствии с данной концепцией, сервис-ориентированный подход позволяет организовать повсеместный доступ к разнородным научным ресурсам и автоматизировать процесс научных исследований,

* Работа выполнена при поддержке фонда РФФИ (грант № 10-07-00176-а) и Президиума РАН (программа П1).

тем самым, повышая производительность исследований и открывая новые возможности для науки в целом.

В настоящее время отсутствуют проработанные подходы к реализации сервис-ориентированных научных сред. Обусловлено это как относительной новизной данного направления, так и рядом технологических проблем, стоящих на пути реализации подобных сред, таких как:

- унификация и обеспечение интероперабельности сервисов на уровне протоколов, интерфейсов, форматов и семантики данных;
- организация безопасного доступа к сервисам, включая защиту передаваемых по сети данных, аутентификацию и авторизацию пользователей, учет использования ресурсов и т.д.;
- снижение технологического барьера для потенциальных разработчиков сервисов за счет упрощения процедур создания и размещения сервисов в сети;
- обеспечение масштабируемости сервисов, в том числе за счет динамического подключения внешних вычислительных ресурсов;
- снижение технологического барьера для потенциальных пользователей сервисов путем реализации проблемно-ориентированных интерфейсов («рабочих пространств»), скрывающих техническую сторону функционирования сервисов и сложность низлежащей вычислительной инфраструктуры;
- создание механизмов публикации, аннотации и оценки качества сервисов, позволяющих пользователю быстро найти интересующий его сервис;
- создание механизмов формирования сообществ пользователей (виртуальных организаций), образующих контексты для разделения сервисов.

Несмотря на имеющийся опыт решения подобных проблем при создании грид-систем, существующие технологии зачастую сложны в использовании и требуют своего пересмотра или поиска новых решений. Сервис-ориентированные научные среды, за счет расширения круга ресурсов и приложений, должны обеспечивать функционирование множества виртуальных сообществ среднего и малого размера. При этом требуется радикально упростить процедуры развертывания соответствующего ПО, формирования сообществ пользователей, разработки сервисов и их размещения в среде. Существующие средства разработки грид-сервисов, такие как Globus Toolkit, изначально ориентированы на разработчиков базовых сервисов грид, сложны в использовании и основаны на обладающих рядом недостатков спецификациях веб-сервисов.

В статье рассматривается распределенная среда MathCloud [2], реализующая концепцию сервис-ориентированных научных сред. Целями среды являются предоставление унифицированного доступа к проблемно-ориентированным вычислительным сервисам и поддержка интеграции данных сервисов при решении прикладных задач. Во главу предлагаемого подхода к реализации среды MathCloud ставятся удобство разработки сервисов, простота доступа к сервисам пользователей и использование открытых технологий.

2. Архитектура среды MathCloud

Архитектура среды MathCloud состоит из нескольких уровней, изображенных на **Рис. 1**.

2.1 Вычислительные ресурсы

Нижний уровень архитектуры MathCloud содержит вычислительные ресурсы, используемые для функционирования сервисов среды. Формально данный уровень не относится к самой среде MathCloud, а формируется из доступных разработчикам и пользователям ресурсов. Унифицированные механизмы интеграции и доступа к вычислительным ресурсам уже сформированы в рамках грид-инфраструктур. Однако стоит отметить, что данные инфраструктуры ориентированы на запуск ограниченных по времени вычислительных заданий и не поддерживают размещение постоянно функционирующих сервисов. В то же время, в рамках коммерческих систем, таких как Amazon EC2, активно развивается модель аренды виртуализованных аппаратных ресурсов (cloud computing), позволяющая размещать на этих ресурсах и сами сервисы.

В рамках среды MathCloud предполагается использование обоих видов вычислительных инфраструктур.

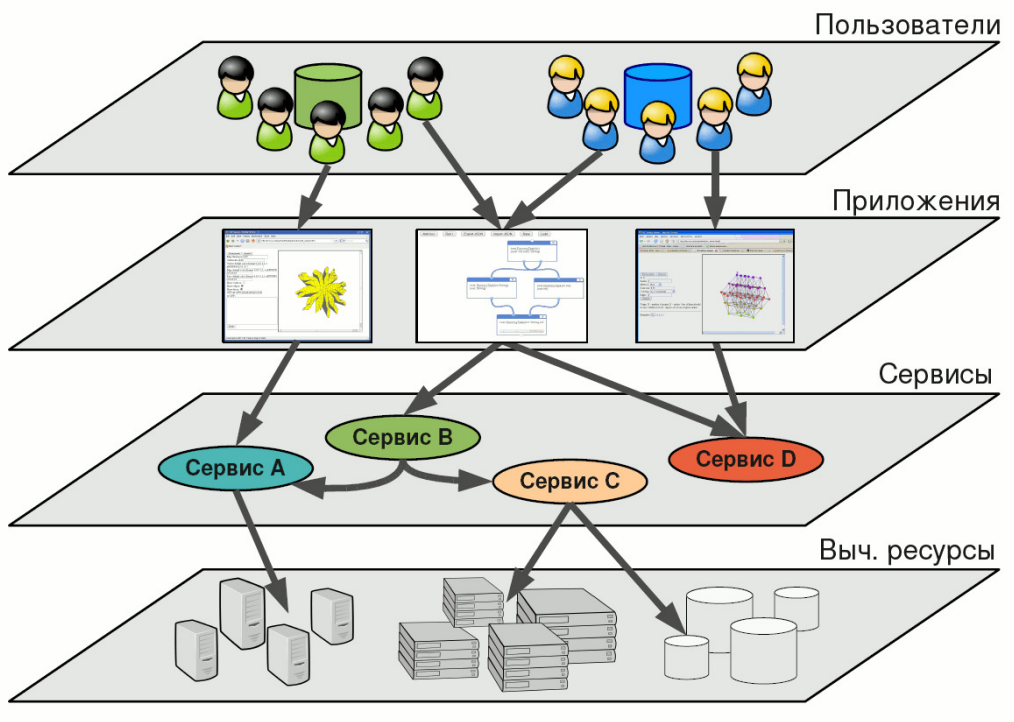

Рис. 1. Уровни архитектуры MathCloud.

Доступ к вычислительным ресурсам может осуществляться с помощью соответствующих интерфейсов менеджеров ресурсов и грид-инфраструктур. Однако, учитывая сложность освоения и использования промежуточного ПО грид-систем, в рамках MathCloud предполагается создание адаптеров, упрощающих взаимодействие сервисов с грид. Данные адаптеры позволят прозрачным для разработчика сервиса способом транслировать запросы к сервису в вычислительные задания, запускаемые в грид. Это также позволяет значительно упростить процесс преобразования в сервисы существующих грид-приложений.

2.2 Сервисы

На уровне сервисов реализуется удаленный программный доступ к некоторой востребованной пользователями среды функциональности. Проблемно-ориентированные сервисы, являющиеся главными компонентами среды, нацелены на решение определенного класса прикладных задач и служат основой для создания приложений. Помимо проблемно-ориентированных сервисов, в среде могут присутствовать системные сервисы, реализующие универсальную функциональность, такую как публикация и поиск сервисов. В дальнейшем под сервисами MathCloud будут подразумеваться проблемно-ориентированные сервисы. Уровень сервисов, как уже было отмечено, опирается на низлежащую вычислительную инфраструктуру, используемую для проведения сложных вычислений или хранения данных.

Остановимся подробнее на модели проблемно-ориентированного сервиса.

Сервис MathCloud представляет собой доступный по сети программный компонент, поддерживающий решение определенного класса задач с помощью соответствующих вычислительных алгоритмов. В соответствии с моделью клиент-сервер, сервис обслуживает приходящие к нему запросы клиентов на решение конкретных задач. Запрос клиента содержит параметризованное описание задачи, формулируемое в виде конечного набора входных параметров. После успешной обработки запроса сервис возвращает клиенту результат, оформленный в виде конечного набора выходных параметров.

Для взаимодействия с сервисом клиенту необходимо знать список входных и выходных параметров сервиса. Данная информация является частью описания сервиса, публикуемого в

общедоступном месте или предоставляемого сервисом по запросу клиента. Описание параметров сервиса определяет форматы сообщений (контракт), которым должны следовать клиент и сервис. Данное описание должно поддерживать машинную интерпретацию, например для валидации сообщений, генерации клиентского кода или реализации динамических вызовов.

Отметим, что описанная модель является довольно общей и может быть, вообще говоря, применена к большому классу сервисов, не обязательно предоставляющих доступ к вычислительным алгоритмам. В частности, это могут быть сервисы доступа к базам данных или сервисы, осуществляющие обработку и преобразование данных. В общем случае, речь может идти о произвольном программном компоненте, оформленном в виде сервиса. Тем не менее, в рамках среды MathCloud предполагается рассмотрение в первую очередь сервисов, нацеленных на решение научных и прикладных вычислительных задач. Рассмотрим характерные особенности вычислительных сервисов.

Во-первых, каждый запрос обрабатывается сервисом независимо от других запросов. Иными словами, результат запроса определяется исключительно значениями передаваемых клиентом входных параметров и не зависит от результатов других запросов. Фактически данная особенность играет роль ограничения, накладываемого на рассматриваемые сервисы. А именно, за рамками описываемой модели сервиса остаются клиент-серверные приложения, в которых требуется поддерживать состояние (сессию) между последовательными запросами клиента к серверу. Исключение из рассмотрения подобных случаев позволяет существенно упростить интерфейс и реализацию сервисов, а также, что самое главное, повысить масштабируемость и отказоустойчивость среды.

Вторая важная особенность вычислительных сервисов заключается в том, что обработка запроса клиента (решение задачи) может потребовать длительных вычислений, в том числе на многопроцессорных вычислительных комплексах или в грид. В этом случае сервис не может вернуть результат сразу же после получения запроса. Обработка подобных запросов должна вестись в асинхронном режиме, путем преобразования поступающих запросов в задания. В ответ на вызов клиенту передается идентификатор соответствующего задания, используя который клиент может опрашивать статус задания и получить результат. Клиент также может получать от сервиса уведомления об изменении статуса задания.

Кроме того, растет количество вычислительных приложений, принимающих на вход или генерирующих большие объемы данных. Для подобных сервисов необходимо организовать эффективную передачу параметров большого размера в виде файлов с использованием соответствующих механизмов передачи данных по сети. В этом случае запрос к сервису или возвращаемый сервисом результат могут содержать не сами значения параметров, а ссылки на соответствующие файлы.

Для упрощения повторного использования и композиции сервисов в рамках различных приложений требуется унификация механизма удаленного доступа к сервисам на уровне протоколов и форматов данных. Для этих целей предлагается использовать архитектурный стиль REST (Representational State Transfer), хорошо зарекомендовавший себя в рамках Web. Для среды MathCloud разработан и подробно описан унифицированный REST-интерфейс (см. раздел 3), который должны реализовывать сервисы среды.

Среда MathCloud является открытой распределенной системой, для участия в которой разработчику сервиса необходимо и достаточно реализовать унифицированный REST-интерфейс. При этом детали реализации сервиса остаются полностью на усмотрение его разработчика. Тем не менее, необходимо наличие готовых средств, упрощающих разработку сервисов, в первую очередь для разработчиков с минимальной квалификацией. Подобные средства особенно актуальны для быстрого преобразования в сервисы существующих приложений. Исходя из этих соображений, была реализована универсальная среда выполнения сервисов (см. раздел 4).

2.3 Приложения

На уровне приложений реализуется доступ пользователей к сервисам среды через проблемно-ориентированные интерфейсы. В качестве стандартной среды выполнения приложений предлагается использовать веб-браузер, что обладает рядом преимуществ: приложение сразу готово к работе без установки на компьютер пользователя, разработчик приложения сохраняет

полный контроль над ним, включая возможность обновлений и учета использования. Так же как в случае сервисов, архитектура не накладывает существенные ограничения на используемые для разработки приложения средства программирования.

Отметим, что реализация некоторого сервиса может быть уже снабжена интерфейсом для работы с сервисом пользователей. Однако в общем случае понятия сервиса и приложения следует различать. Если сервис предоставляет программный интерфейс для приложений, то приложение предоставляет интерфейс конечному пользователю. Таким образом, один и тот же сервис может использоваться в рамках нескольких приложений. И наоборот - в рамках одного приложения могут быть задействованы сразу несколько сервисов. Остановимся на последнем моменте подробнее.

Ключевой функцией предлагаемой сервис-ориентированной среды является поддержка объединения или композиции сервисов, позволяющая пользователям среды «собирать» из существующих сервисов новые приложения и, что особенно важно, новые сервисы, развивая, тем самым, среду. В общем случае, композиция сервисов может осуществляться с применением произвольных средств программирования, например скриптовых языков. Однако, также как и в случае с разработкой сервисов, в рамках MathCloud предусмотрены готовые средства, упрощающие композицию сервисов и доступные пользователям с минимальной квалификацией. Для поддержки интеграции сервисов среды при решении прикладных задач реализованы редактор композитных приложений и система управления сценариями на основе workflow-подхода (см. раздел 5).

2.4 Пользователи

На уровне пользователей сконцентрирован социальный аспект среды. Здесь решаются задачи, связанные с формированием и обеспечением функционирования сообществ пользователей. Предполагается, что данные сообщества, как правило, будут формироваться вокруг той или иной области исследований, аналогично виртуальным организациям в грид. Идеальной средой для формирования таких сообществ представляются Web-порталы с функциональностью социальных сетей, учитывающие научную специфику проекта. Будучи созданными, подобные сообщества могут образовывать контексты для разделения ресурсов и сервисов. Во-первых, в рамках сообщества могут коллективно вестись публикация, каталогизация и учет сервисов, относящихся к тематике данного сообщества. Во-вторых, аналогично грид, доступ к сервисам может регламентироваться на основе принадлежности пользователя к определенному сообществу. В-третьих, интерфейсы приложений могут быть интегрированы в портал сообщества, предоставляя удобный доступ к сервисам среды. Данный уровень архитектуры MathCloud является в данный момент наименее проработанным, поскольку текущие усилия сосредоточены на нижних, технологических уровнях среды.

3. Интерфейс сервиса MathCloud

Описанная архитектура носит общий характер и может быть реализована с помощью различных технологий построения распределенных систем. Выбор той или иной технологии зависит от ряда факторов, таких как распространенность, удобство разработки и открытость исходного кода. В рамках научной сервис-ориентированной среды, допускающей участие различных лиц и организаций, очень важно использование открытых стандартов и наличие нескольких независимых реализаций базовой технологии. Важно также, чтобы используемая технология как можно ближе соответствовала описанной выше модели проблемно-ориентированного вычислительного сервиса.

В настоящее время доминирующей технологией для построения сервис-ориентированных систем являются Web-сервисы на основе протокола SOAP и спецификаций WS-*, которые будем далее называть WS-сервисами. Спецификации WS-сервисов являются открытыми стандартами, изначально ориентированными на реализацию сервис-ориентированной архитектуры. Наконец, что немаловажно, данные технологии активно поддерживаются крупными компаниями-разработчиками, что привело к повсеместному их распространению и появлению множества реализаций, в том числе и с открытым кодом.

Распространенной критикой WS-сервисов является их чрезмерная сложность и некорректное использование протокола HTTP. При этом преимущества от использования WS-сервисов заметны только в определенном классе приложений, ориентированных на поддержку сложных бизнес-процессов, встречающихся в корпоративных и государственных системах и слабо распространенных в технических и научных проектах. Это привело к появлению альтернативных, более простых в использовании подходов к реализации Web-сервисов, основанных на прямом использовании протокола HTTP.

В 2000 году Рой Филдинг, один из главных разработчиков протокола HTTP и других спецификаций Web, опубликовал диссертацию [3] с описанием эталонной модели архитектуры Web. Данная модель или архитектурный стиль, получивший название Representational State Transfer (REST), содержит ряд ключевых принципов, определенных в виде накладываемых на архитектуру ограничений. Центральными понятиями REST являются понятия ресурса, идентификатора и представления ресурса. В рамках REST вводятся следующие ограничения на архитектуру системы: клиент-серверная архитектура, хранение состояния приложения на стороне клиента, кэширование ответов на запросы, унифицированный интерфейс доступа к ресурсам, многоуровневая архитектура. Данные ограничения позволяют обеспечить такие свойства архитектуры Web, как масштабируемость, расширяемость и открытость.

Благодаря унифицированному интерфейсу доступа к ресурсам, использованию открытых стандартов (HTTP, URI) и наличию множества проверенных временем реализаций (Web-серверы, библиотеки для работы с HTTP для всех современных языков программирования), REST обеспечивает максимальную свободу для независимой разработки Web-сервисов и соответствующих клиентских приложений. Немаловажным обстоятельством является простота разработки REST-сервисов в сравнении с другими рассмотренными технологиями. Все это позволяет максимально расширить как круг потенциальных разработчиков, так и пользователей сервисов, обеспечив при этом совместимость различных реализаций и широкое повторное использование сервисов. Это привело к широкому распространению Web-сервисов на основе стиля REST (т.н. RESTful Web-сервисов), особенно в рамках Web 2.0 приложений.

Исходя из вышеописанного, в качестве базовой платформы для организации удаленного доступа к сервисам MathCloud предлагается использовать протоколы и технологии Web, основанные на архитектурном стиле REST. Разработан и подробно описан REST-интерфейс сервиса MathCloud [4], использующий в качестве протокола доступа к сервисам протокол HTTP. Разработанный интерфейс учитывает характерные особенности вычислительных сервисов, такие как длительная обработка запросов и передача больших объемов данных в виде файлов. Также в соответствии с сервис-ориентированным подходом разработанный интерфейс поддерживает интроспекцию, т.е. получение информации о сервисе. Выбранные базовые технологии допускают независимые реализации описанного интерфейса на различных языках программирования. При этом детали реализации сервиса остаются полностью на усмотрение его разработчика.

В соответствии с принципами REST, интерфейс сервиса MathCloud образован совокупностью идентифицируемых при помощи URI ресурсов, поддерживающих стандартные методы протокола HTTP (Табл. 1).

Таблица 1. Ресурсы и методы интерфейса сервиса MathCloud

Ресурс	GET	POST	DELETE
Сервис SERVICE_URI	Получение описания сервиса	Запрос к сервису	—
Задание JOB_URI	Получение статуса и результатов задания	—	Отмена задания, удаление результатов задания
Файл результата задания FILE_URI	Загрузка файла	—	—
Сервер SERVER_URI	Получение списка сервисов, размещенных на сервере	—	—

Ресурс-сервис поддерживает два метода. Метод GET возвращает клиенту представление данного ресурса, содержащее описание сервиса. Метод POST служит для отправки сервису нового запроса. В теле запроса клиент передает набор значений входных параметров задачи. В ответ сервис создает новый ресурс-задание, являющийся подчиненным по отношению к ресурсу-сервису, и возвращает идентификатор ресурса-задания и его текущее представление клиенту.

Ресурс-задание поддерживает методы GET и DELETE. Метод GET возвращает представление данного ресурса, содержащее информацию о текущем статусе задания. Данная информация должна обязательно включать состояние задания, которое может принимать следующие значения:

- WAITING - выполнение задания еще не началось;
- RUNNING - задание выполняется;
- DONE - задание выполнено успешно;
- FAILED - выполнение задания завершилось ошибкой.

Если выполнение задания завершено успешно (состояние DONE), то в представление ресурса-задания также включается результат задания в виде набора значений выходных параметров. Часть значений выходных параметров может содержать идентификаторы ресурсов-файлов.

В случае если в процессе выполнения задания стали известны значения части выходных параметров, данные значения могут включаться в представление ресурса-задания еще до окончания выполнения задания.

Метод DELETE ресурса-задания позволяет клиенту отменить выполнение запроса или, если выполнение уже завершено, удалить результаты задания. После вызова DELETE данный ресурс-задание, а также подчиненные ему ресурсы-файлы, перестают существовать.

Ресурс-файл представляет собой часть результата задания и поддерживает метод GET для получения содержимого файла клиентом.

Дополнительный ресурс-сервер предназначен для случаев, когда в рамках одного HTTP-сервера размещено несколько алгоритмических сервисов. В этих случаях может быть полезным получение списка сервисов, размещенных на сервере. Для этих целей зарезервирован метод GET. Ресурс-сервер в данном случае является родительским по отношению к ресурсам-сервисам.

Описанный интерфейс поддерживает обработку запросов как в синхронном, так и асинхронном режиме. Действительно, если результат запроса может быть сразу возвращен клиенту, то он передается внутри возвращаемого в ответ клиенту представления ресурса-задания с указанием состояния DONE. Если же для обработки запроса требуется время, то это указывается внутри возвращаемого клиенту представления ресурса-задания с помощью указания соответствующего состояния задания (WAITING или RUNNING). В этом случае клиент использует переданный URI ресурса-задания для дальнейшего опроса состояния задания и получения результата.

4. Среда выполнения сервисов

Наличие готового и простого в использования инструмента для создания сервисов очень важно с точки зрения расширения круга потенциальных разработчиков сервисов. Для упрощения процесса разработки сервисов MathCloud была реализован контейнер сервисов Everest. Данный контейнер представляет собой универсальную среду выполнения сервисов, реализующую описанный выше интерфейс сервиса.

Универсальность контейнера заключается в том, что он позволяет легко, во многих случаях без написания программного кода, преобразовать в сервис MathCloud широкий спектр существующих приложений. Кроме того, реализованный в контейнере механизм адаптеров позволяет подключать произвольные реализации обработчиков запросов. В частности, путем преобразования запроса к сервису в вычислительное задание, на уровне адаптера может быть реализован доступ к вычислительной инфраструктуре.

Everest реализован на базе библиотеки Jersey, являющейся в свою очередь открытой эталонной реализацией спецификации JAX-RS (Java API for RESTful Web Services). На **Рис. 2** изо-

бражена архитектура Everest. Взаимодействие с клиентами осуществляется с помощью встроенного в контейнер Web-сервера Jetty. Входящие HTTP-запросы передаются библиотеке Jersey и, затем, реализации Everest. Сопряжение между Jersey и Everest осуществляется с помощью четырех Java-классов (ServerResource, ServiceResource, JobResource и FileResource), которые являются реализациями соответствующих ресурсов из REST-интерфейса сервиса MathCloud.

Контейнер осуществляет обработку запросов клиентов в соответствии с конфигурационной информацией. Компонент ServiceManager хранит список сервисов, размещенных на сервере, и их конфигурацию. Данная информация считывается при запуске сервера из конфигурационных файлов. Конфигурация каждого сервиса состоит из двух частей:

- внешнее описание сервиса, передаваемое клиентам (текстовая аннотация сервиса, описания входных и выходных параметров);
- внутренняя конфигурация сервиса, содержащая информацию о реализации сервиса (адаптер и его конфигурационные параметры).

Компонент JobManager осуществляет управление обработкой запросов к сервисам. Запросы преобразуются в задания и размещаются в очереди, откуда затем извлекаются конфигурируемым пулом потоков-обработчиков. При выполнении задания поток-обработчик вызывает тот или иной адаптер, в соответствии с внутренней конфигурацией сервиса.

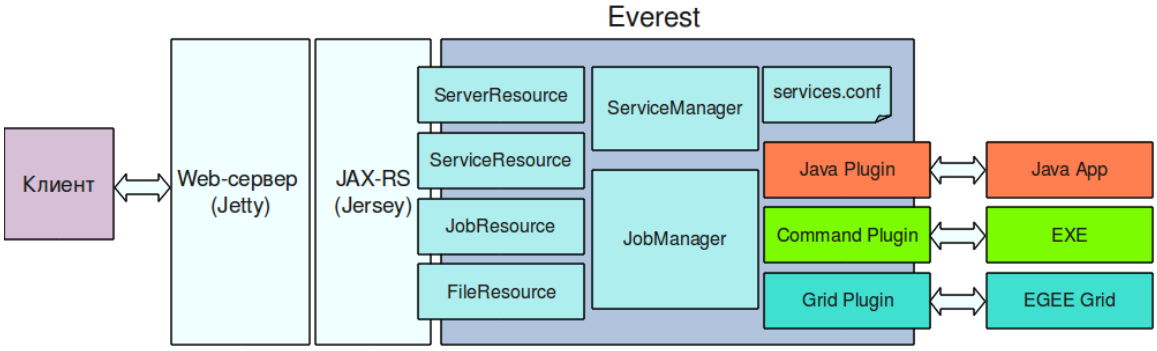

Рис. 2. Архитектура Everest.

Компоненты, реализующие обработку запросов к сервисам (заданий), оформляются в виде подключаемых адаптеров. Каждый адаптер реализует стандартный программный интерфейс, с помощью которого контейнер передает адаптеру параметры запроса, контролирует состояние выполнения задания и получает результаты. Реализация адаптера, как правило, преобразует полученный запрос в вызов внешнего приложения. После успешного завершения приложения адаптер сохраняет полученные результаты и изменяет состояние задания на DONE.

В настоящее время реализовано три универсальных адаптера, поддерживающих интеграцию следующих типов приложений:

- приложение с интерфейсом командной строки (адаптер Command);
- приложение на языке Java (адаптер Java);
- грид-задание, запускаемое в инфраструктуре EGEE (адаптер Grid).

Адаптер Command осуществляет преобразование запроса к сервису в запуск команды в отдельном процессе операционной системы. Во внутренней конфигурации сервиса необходимо указать запускаемую команду в специальном формате, содержащем информацию о том, каким образом параметры сервиса отображаются в аргументы команды и внешние файлы.

Адаптер Java осуществляет вызов заданного Java-класса в рамках текущей виртуальной машины, передавая в вызове параметры запроса к сервису. Используемый Java-класс должен реализовывать стандартный программный интерфейс. Во внутренней конфигурации сервиса требуется указать имя соответствующего Java-класса.

Адаптер Grid осуществляет преобразование запроса к сервису в запуск вычислительного задания в грид-инфраструктуре EGEE, основанной на ПО gLite. Данный адаптер может использоваться как для преобразования в сервис существующего грид-приложения, так и для переноса существующей реализации сервиса в грид, например, из соображений производительности и масштабируемости. Во внутренней конфигурации сервиса требуется указать виртуальную ор-

ганизацию грид, путь к файлу с описанием грид-задания в формате JD, а также информацию об отображении параметров сервиса в аргументы и файлы грид-задания.

Отметим, что адаптеры Command и Grid позволяют преобразовывать существующие приложения в сервисы без разработки дополнительных программных компонентов. Данная возможность позволяет быстро преобразовать в сервисы широкий спектр существующих приложений, не обладая при этом навыками программирования.

Каждый размещенный в контейнере сервис становится доступным через стандартный интерфейс сервиса MathCloud. Кроме того, Everest поддерживает дополнительную HTML-версию данного интерфейса, позволяющую пользователям непосредственно обращаться к сервисам через веб-браузер.

5. Композиция сервисов и система управления сценариями

Для поддержки интеграции сервисов среды MathCloud при решении прикладных задач реализована система управления сценариями на основе workflow-подхода [5]. Данная система реализует описание, хранение, выполнение и публикацию сценариев совместного использования сервисов среды. Сценарии описываются в виде ориентированных ациклических графов при помощи визуального редактора. Описанный сценарий может быть затем преобразован в новый сервис среды и запущен на выполнение путем вызова данного сервиса. Предлагаемый подход позволяет скрыть от пользователя детали реализации вызовов сервисов и передачи данных между ними, оставив только необходимость правильного соединения сервисов друг с другом. Таким образом, многие задачи, решение которых сводится к компоновке типовых сервисов, становятся доступными для пользователей, не обладающих навыками распределенного программирования. Графическое представление сценария позволяет сделать наглядными связи между сервисами. Кроме того, такое представление позволяет быстро вносить изменения в уже работающие сценарии - такие, как добавление новых блоков сценария или замена отдельных блоков другими, получая на выходе новые сервисы с новыми параметрами.

Разработанная система имеет распределенную клиент-серверную архитектуру. Клиентская часть системы включает редактор сценариев, а серверная - сервис управления сценариями и среду выполнения сценариев.

5.1 Редактор сценариев

Редактор сценариев реализует графический интерфейс, предназначенный для взаимодействия с системой пользователей. Редактор является клиентом сервиса управления сценариями и может функционировать на любой машине в сети. Основными функциями редактора являются просмотр и редактирование сценариев. Созданный с помощью редактора сценарий может быть сохранен и запущен на выполнение на серверной стороне. При этом в редакторе отображается состояние выполнения сценария.

Редактор сценариев реализован в виде веб-приложения на языке JavaScript. Благодаря этому редактор может использоваться без предварительной установки на любом компьютере, где установлен современный веб-браузер. Редактор создан с широким использованием технологий AJAX. В его основе лежат такие JavaScript-библиотеки, как WireIt, YUI и InputEx. Это позволило сделать интерфейс редактора легковесным, интуитивным и удобным для пользователя.

На **Рис. 3** изображен общий вид редактора сценариев. Правая часть редактора содержит список доступных сервисов и прочих элементарных блоков, из которых пользователь может компоновать сценарий. Верхняя часть представляет собой главное меню, предоставляющее доступ к основным операциям со сценариями - открытие, сохранение, запуск и т.д. В центральной части редактора размещается графическое представление сценария.

Сценарий представлен в виде ориентированного ациклического графа, вершинами которого являются элементарные блоки сценария, а ребрами - соединения, задающие потоки данных между блоками. Каждый блок имеет набор входов и выходов, отображаемых в виде круглых портов соответственно сверху и снизу блока. Блок реализует некоторую логику обработки поступающих на вход данных. Передача данных между блоками реализуется путем соединения

выхода одного блока с входом другого. С каждым выходом и выходом связан определенный тип данных. При соединении блоков проверяется совместимость типов входа и выхода.

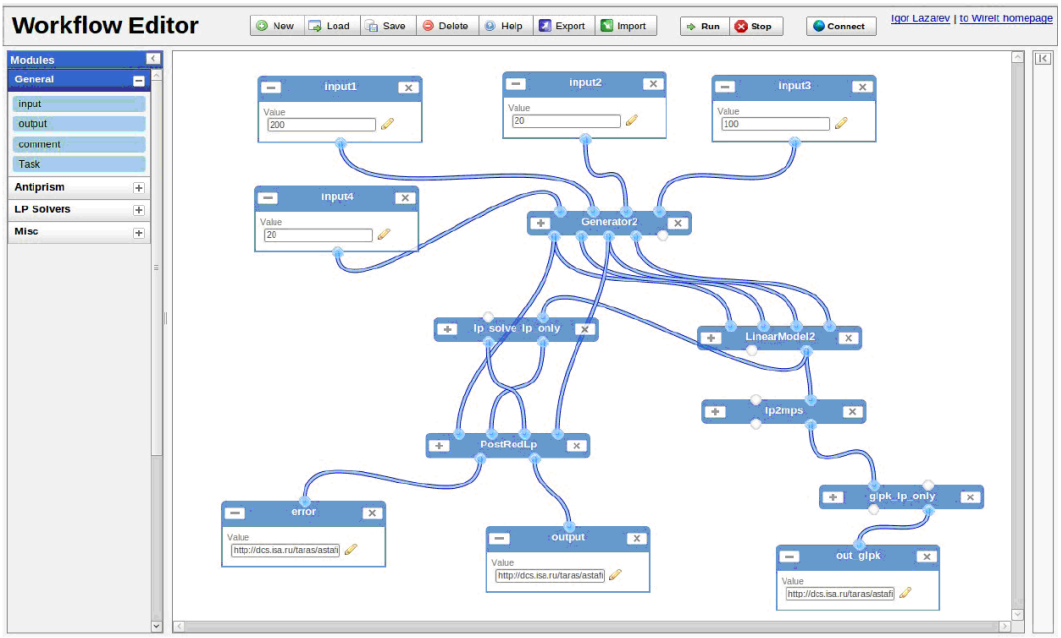

Рис. 3. Общий вид редактора сценариев.

В настоящее время реализованы следующие типы блоков сценария:

- Input - служит для задания значения входного параметра сценария;
- Output - служит для отображения значения выходного параметра сценария;
- Service - осуществляет вызов заданного сервиса MathCloud;
- JavaScript - осуществляет выполнение заданного кода на языка JavaScript;
- Python - осуществляет выполнение заданного кода на языке Python;
- Comment - служит для размещения комментариев к сценарию.

Добавление в сценарий сервиса MathCloud осуществляется путем создания нового блока Service и ввода пользователем URI требуемого сервиса. Редактор запрашивает описание данного сервиса, откуда извлекает информацию о количестве, типах и именах входных и выходных параметров сервиса. После чего динамически формируются соответствующие входы и выходы блока, а в заголовке блока отображается имя сервиса. В случае неудачи при соединении с сервисом блок окрашивается в красный цвет.

Для упрощения создания блоков Input и Output, редактор позволяет автоматически создавать данные блоки путем указания пользователем требуемого входа или выхода блока.

Блоки JavaScript и Python позволяют увеличить гибкость при описании сценария путем внедрения в него фрагментов кода (скриптов). Скрипт оформляется в виде функции, сигнатура которой используется для динамической генерации входов и заголовка блока. Выход блока соответствует возвращаемому значению функции. Выполнение скрипта осуществляется на серверной стороне средой выполнения сценариев.

Редактор поддерживает передачу и сохранение созданного сценария на стороне сервиса управления сценариями. Также с помощью редактора можно просматривать список сохраненных сценариев, открывать сохраненный сценарий, удалять сценарий, осуществлять импорт и экспорт описания сценария в формате JSON.

Важной функцией редактора является запуск сценария на выполнение. Для этого необходимо задать значения всех входных параметров сценария с помощью соответствующих блоков Input. После нажатия пользователем кнопки Run, редактор производит вызов сервиса-сценария с заданными входными параметрами. Далее редактор осуществляет периодический опрос состояния запущенного задания, содержащего информацию о состоянии отдельных блоков сценария. Данная информация отображается пользователю путем окрашивания блоков сценария в цвет, соответствующий текущему состоянию. После успешного завершения сценария, значения

выходных параметров сценария отображаются в соответствующих блоках Output. Каждый запущенный сценарий имеет уникальный URI, используя который можно в любой момент открыть в редакторе выполняющийся сценарий. Данная возможность особенно важна в тех случаях, когда сценарий выполняется в течение длительного времени.

5.2 Сервис управления сценариями

Сервис управления сценариями реализует удаленный программный интерфейс для хранения сценариев, созданных с помощью редактора. В соответствии с сервис-ориентированным подходом, сервис управления сценариями также обеспечивает развертывание каждого сохраненного пользователем сценария в виде нового сервиса среды MathCloud.

Удаленный интерфейс сервиса управления сценариями реализован на основе протокола HTTP и архитектурного стиля REST, что позволяет удобным образом взаимодействовать с сервисом из редактора сценария. В соответствии с принципами REST интерфейс сервиса образован совокупностью идентифицируемых при помощи URI ресурсов, поддерживающих стандартные методы протокола HTTP (Табл. 2).

Таблица 2. Ресурсы и методы интерфейса сервиса управления сценариями.

Ресурс	GET	POST	PUT	DELETE
Сервис управления сценариями WFMS_SERVICE_URI	Получение списка хранимых сценариев	Создание нового сценария	—	—
Сценарий WF_URI	Получение описания сценария	—	Обновление сценария	Удаление сценария
Сервис сценария WF_SERVICE_URI	Получение описания сервиса	Запуск сценария	—	—
Прокси WFMS_PROXY_URI	Получение описания сервиса с указанным URI	—	—	—

Сервис сценария реализует унифицированный REST-интерфейс сервиса MathCloud (см. раздел 3). Таким образом, с точки зрения клиентов сервис сценария ничем не отличается от обычных сервисов и запуск сценария на выполнение осуществляется путем отправки запроса к его сервису. Это позволяет, например, легко использовать существующий сценарий в качестве одного из элементов при создании нового сценария.

5.3 Среда выполнения сценариев

Среда выполнения сценариев осуществляет обработку запросов к сервисам-сценариям. Каждый подобный запрос приводит к запуску нового экземпляра соответствующего сценария. Среде выполнения сценариев передается описание сценария и значения его входных параметров. Среда осуществляет интерпретацию описания сценария, производит указанные в сценарии действия (в том числе вызовы внешних сервисов), отслеживает состояние выполнения сценария и возвращает результаты его выполнения. Основой среды выполнения является разработанная библиотека WorkflowRuntime на языке Java, поддерживающая выполнение произвольных workflow-сценариев, заданных в виде ориентированных ациклических графов.

6. Заключение

В статье рассмотрены принципы реализации и основные компоненты сервис-ориентированной научной среды MathCloud. Архитектура среды основана на представлении сервиса в виде функции с заданным набором входных и выходных параметров и применении стиля REST для описания унифицированного интерфейса сервиса. Разработанный интерфейс сервиса MathCloud учитывает характерные особенности вычислительных сервисов, такие как

длительная обработка запросов и передача больших объемов данных в виде файлов. Для упрощения процесса разработки сервисов MathCloud реализован универсальный контейнер сервисов, позволяющий быстро, без написания дополнительного программного кода, преобразовать в сервисы широкий спектр существующих приложений. Реализованный в контейнере механизм адаптеров позволяет подключать произвольные реализации обработчиков запросов к сервису. В частности, путем трансляции запроса в вычислительное задание, на уровне адаптера может быть реализован доступ к различным видам вычислительных ресурсов. Данный механизм позволяет упростить процесс преобразования в сервисы существующих параллельных и грид-приложений, а также перенос реализации сервиса на новую вычислительную инфраструктуру. Для поддержки интеграции сервисов среды при решении прикладных задач реализованы графический редактор и система управления сценариями на основе workflow-подхода.

Одним из важных аспектов архитектуры среды MathCloud, оставшимся за рамками данной статьи, является вопрос обеспечения безопасности. Предполагается, что многие из сервисов MathCloud будут открыты для публичного доступа. Тем не менее, очевидно, что в ряде случаев будут необходимы ограничения круга пользователей сервиса, и что жизнеспособность предлагаемой среды будет зависеть от наличия полноценного механизма безопасности. Сюда входят аутентификация пользователей и сервисов, авторизация пользователей и контроль доступа к сервисам, защита передаваемых по сети данных и делегация прав доступа. В настоящее время ведется разработка механизма безопасности среды MathCloud, основанного на применении протокола HTTPS и цифровых сертификатов. Также предусмотрен упрощенный способ аутентификации пользователей при помощи публичных провайдеров учетных записей и стандарта OpenID.

Другим важным компонентом среды MathCloud, работа над которым еще не завершена, является каталог сервисов, поддерживающий публикацию, поиск и мониторинг сервисов среды.

Данные вопросы будут подробно рассмотрены в рамках дальнейших работ.

Литература

1. Foster I. Service-Oriented Science // Science. 2005. Vol. 308, No. 5723. P. 814-817.
2. Астафьев А.С., Афанасьев А.П., Лазарев И.В., Сухорослов О.В., Тарасов А.С. Научная сервис-ориентированная среда на основе технологий Web и распределенных вычислений. // Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность: Труды Всероссийской суперкомпьютерной конференции (21-26 сентября 2009 г., г. Новороссийск). Москва: Изд-во МГУ, 2009. С. 463-467.
3. Fielding, R.T. Architectural styles and the design of network-based software architectures. PhD Dissertation. Dept. of Information and Computer Science, University of California, Irvine, 2000.
4. Сухорослов О.В. Унифицированный интерфейс доступа к алгоритмическим сервисам в Web. // Проблемы вычислений в распределенной среде / Под ред. С.В. Емельянова, А.П. Афанасьева. Труды ИСА РАН, Т. 46. Москва: КРАСАНД, 2009. С. 60-82.
5. Лазарев И.В., Сухорослов О.В. Реализация распределенных вычислительных сценариев в среде MathCloud. // Проблемы вычислений в распределенной среде / Под ред. С.В. Емельянова, А.П. Афанасьева. Труды ИСА РАН, Т. 46. Москва: КРАСАНД, 2009. С. 6-23.