Интеграция инструментария IARnet с технологией ICE[1]

А.П. Афанасьев, В.В. Волошинов, О.В. Сухорослов

Аннотация. В статье описываются архитектура и реализация экспериментальной версии инструментария IARnet, основанной на технологии Ice. В качестве примера использования инструментария рассматривается реализация высокоуровневого сервиса обработки геоданных.

Введение

Инструментарий IARnet [1.2] предоставляет средства для интеграции программных ресурсов в распределенную вычислительную среду типа Grid [3], обеспечивающие их координированное совместное использование. IARnet ориентирован на трансформацию существующих научных приложений и алгоритмических ресурсов (математических пакетов, различных численных «решателей», имитационных моделей и т.п.) в удаленно доступные сервисы с четко описанными интерфейсами. Обеспечивая удаленный доступ и совместное использование приложений, IARnet дополняет возможности современной Grid-инфраструктуры.

Grid-системы первого поколения, такие как Large Hadron Collider Grid (LCG), ориентированы на крупные научные проекты и виртуальные организации с большим числом участников. В настоящее время происходит формирование второго поколения Grid-систем, которое обобщает идеи совместного использования вычислительных ресурсов на широкий круг ресурсов и приложений, предлагая универсальную инфраструктуру для научной кооперации [4,5]. Данная инфраструктура базируется на сервис-ориентированном подходе [6]: пользователи преобразуют свои ресурсы в удаленно доступные сервисы, которые могут быть обнаружены и использованы другими пользователями в своих приложениях для решения задач. Под ресурсами здесь подразумеваются не столько суперкомпьютеры и хранилища данных, сколько различные научные приложения, средства анализа данных, приборы и т.п.

При таком расширении круга ресурсов и приложений системы второго поколения должны обеспечивать функционирование множества виртуальных организаций среднего и малого размера. При этом требуется радикально упростить процедуры развертывания соответствующего ПО, формирования виртуальных организаций, разработки сервисов и их размещения в Grid. Сложность освоения существующих Grid-технологий сужает круг потенциальных разработчиков сервисов. Отсутствие удобных средств разработки также увеличивает время, необходимое для трансформации существующего ресурса в сервис.

Главной целью инструментария IARnet является создание высокоуровневых средств, упрощающих разработку сервисов и приложений, а также не требующих от разработчика

[1] Работа выполнена при поддержке фонда РФФИ (грант № 05-07-90182-в), Президиума РАН (программа фундаментальных исследований 15П), Федерального агентства по науке и инновациям (государственный контракт № 02.514.11.4053) и Совета по грантам Президента РФ (грант МК-3128.2007.9).

глубоких знаний технологий промежуточного программного обеспечения (ППО, middleware). Подобные средства позволяют разработчику сосредоточиться на решении стоящих перед ним прикладных задач.

Инструментарий IARnet содержит высокоуровневые средства для переноса программных компонентов, реализованных на языках Java и C++, в распределенную вычислительную среду и для доступа к данным ресурсам из Java-приложений, а именно:

- интерфейс прикладного программирования Java Client API, который скрывает от программиста детали реализации удаленных вызовов сервисов, т.е. нейтральный по отношению к ППО;

- средства разработки сервисов на языках Java и C++, также нейтральные по отношению к используемому для вызова сервиса ППО;

- несколько реализаций сетевого уровня IARnet, основанных на различных технологиях ППО (CORBA, Web-сервисы, Ice), и соответствующие среды размещения сервисов (т.н. контейнеры);

- несколько сервисов общего назначения (служб), таких как информационная служба и служба управления сценариями.

Главными преимуществами IARnet являются простая и легкая в использовании модель программирования, а также возможность использования на сетевом уровне любых реализаций механизма удаленных вызовов. В то же время, в IARnet отсутствует такие функциональности, как

- полноценный язык описания интерфейсов с поддержкой произвольных типов данных;

- механизм безопасности с поддержкой конфиденциальности, целостности данных, аутентификации и авторизации;

- средства конфигурации, развертывания и администрирования сервисов.

В настоящей статье рассматривается экспериментальная версия инструментария с названием IARnet4Ice, которая устраняет указанные недостатки и расширяет функциональность инструментария путем интеграции с технологией ППО Ice. В качестве примера использования IARnet4Ice рассматривается реализация универсального сервиса обработки геоданных.

1. Архитектура и реализация экспериментальной версии IARnet4Ice

Реализация отсутствующей функциональности, нейтральным по отношению к ППО образом в соответствии с оригинальной архитектурой IARnet, потребовала бы значительных усилий. В этом случае пришлось бы либо полностью реализовать эту функциональность с нуля, либо обеспечить интероперабельность различных реализаций, уже присутствующих в технологиях ППО. Более реалистичными вариантами являются использование одной технологии ППО или ограничение используемых языков программирования только Java. Оба этих варианта подразумевают тесную интеграцию инструментария с одной технологической платформой и существенную переработку оригинальной архитектуры IARnet.

Поскольку научные приложений и алгоритмические ресурсы, которые требуется преобразовать в сервисы, могут быть написаны на различных языках программирования, был выбран первый вариант. А именно, из трех технологий ППО, используемых изначально в IARnet, за основу для экспериментальной версии была выбрана технология Ice [7,8]. Данная технология обладает рядом достоинств, делающих ее привлекательной для реализации распределенных научных приложений и вычислительных сред:

- простота и удобство использования в сравнении с другими технологиями ППО;

- объектно-ориентированный стиль программирования;

- поддержка многих популярных языков программирования и операционных систем;

- возможность использования различных протоколов на транспортном уровне;

- реализация всего спектра моделей взаимодействия: синхронных, асинхронных и односторонних вызовов, а также рассылки сообщений;

- поддержка многопоточных клиентов и серверов;

- поддержка шифрования с открытым ключом и протокола SSL для реализации безопасного взаимодействия клиентов и серверов;

- наличие дополнительных сервисов с богатой функциональностью.

В [9] приводится детальное описание технологии Ice и ее сравнение с технологиями CORBA и Web-сервисов.

1.1. Архитектура IARnet4Ice

Архитектура IARnet4Ice представляет собой рефакторинг оригинальной архитектуры IARnet и вводит новые понятия, которые согласуют ее со структурой современных Grid-систем.

1.1.1. Модель Grid-среды

Распределенная вычислительная среда на основе IARnet4Ice соответствует структуре современных Grid-систем. Данные системы обеспечивают функционирование *виртуальных организаций* (ВО) – динамических совокупностей пользователей и организаций, которые разделяют свои ресурсы для достижения общих целей, совместного решения задач и т.п. Каждый пользователь Grid является членом одной или нескольких ВО. Доступ пользователей к ресурсам осуществляется с помощью соответствующих сервисов. Каждый ресурс имеет политику доступа, определяемую его владельцем и основанную обычно на членстве пользователя в определенных ВО.

Термин *Grid-сайт* используется для описания набора сервисов и ресурсов, установленных и управляемых одной группой лиц. Grid-система состоит их множества автономных сайтов, каждый из которых обеспечивает функционирование набора серверов с размещенными на них сервисами и ресурсами. Доступ пользователей и сервисов Grid к ресурсам сайта организуется через унифицированные интерфейсы.

1.1.2. Базовые понятия

В соответствии с сервис-ориентированной архитектурой и описанной выше моделью Grid-среды, вводятся понятия сервисов и сайтов IARnet.

Сервисом IARnet называется абстрактный элемент распределенной вычислительной среды, предоставляющий клиентам некоторую функциональность. Данная функциональность реализуется некоторым научным приложением или алгоритмическим ресурсом. Сервис действует в роли посредника между клиентами и данным ресурсом, делегируя обработку поступающих вызовов реализации ресурса. Понятие сервиса соответствует понятию *информационно-алгоритмического ресурса* из оригинальной версии IARnet, а реализация сервиса эквивалентна *агенту доступа* IARnet.

Удаленный доступ к сервису обеспечивается одним или несколькими Ice-объектами. Интерфейсы этих объектов образуют интерфейсы сервиса. У сервиса есть *главный Ice-объект*, с вызова которого клиент по умолчанию начинает взаимодействие с сервисом. Главный интерфейс данного объекта называется *главным интерфейсом сервиса*. В простых случаях сервис реализуется с помощью одного Ice-объекта, имеющего один интерфейс. Примером сервиса, функционирование которого обеспечивается множеством Ice-объектов, является сервис-фабрика.

Сайтом IARnet (Рис.1) называется совокупность сервисов IARnet, размещенных и управляемых одной группой лиц. Каждый сайт реализует унифицированный интерфейс, используемый клиентами для аутентификации и получения доступа к сервисам. Физически сайт может размещаться как на нескольких серверах (типичный Grid-сайт), так и на одной машине. Это является отличительной особенностью IARnet4Ice. В то время как типичные Grid-сайты сложны в установке и обычно управляются организацией, то сайт IARnet может быть установлен на рабочей станции и управляться одним человеком.

Клиентом IARnet называется Ice-приложение, которое осуществляет доступ к сервисам IARnet от имени пользователя или другого сервиса.

Архитектура IARnet4Ice состоит из двух основных частей (Рис.2): *сайта IARnet* на серверной стороне и интерфейса прикладного программирования *IARnet4Ice Client API* на клиентской стороне. Сайт реализует унифицированный удаленный интерфейс и предоставляет безопасную среду размещения сервисов. Client API представляет собой простой в использовании API для доступа к сайтам и сервисам из клиентских приложений.

Работая поверх Ice, IARnet4Ice не скрывает от разработчика API Ice и не модифицирует существенно модель программирования Ice. Это означает, что, в отличие от оригинальной

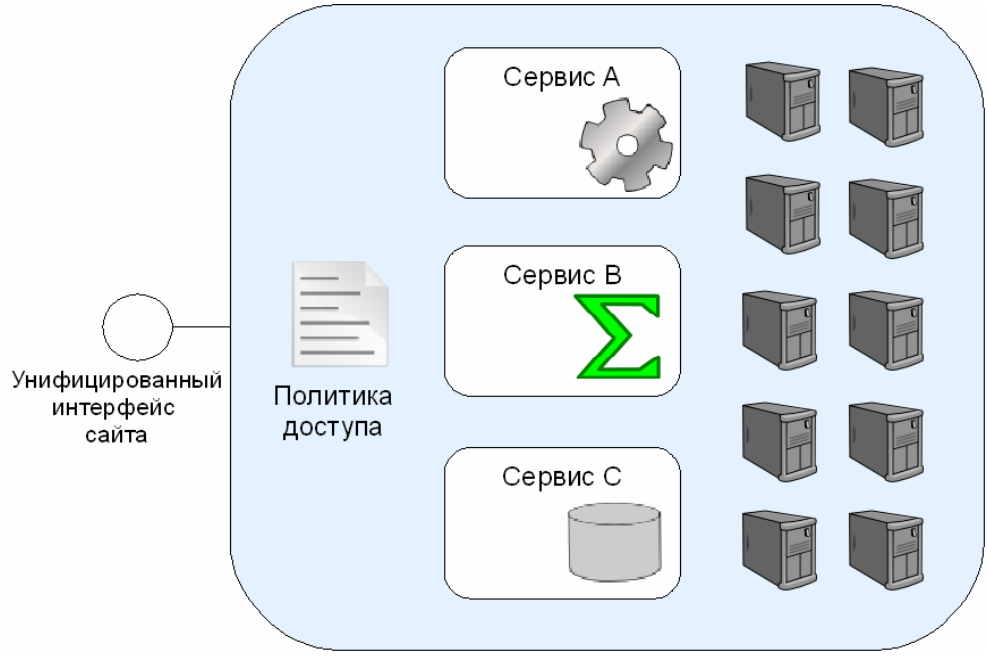

Рис.1. Сайт IARnet

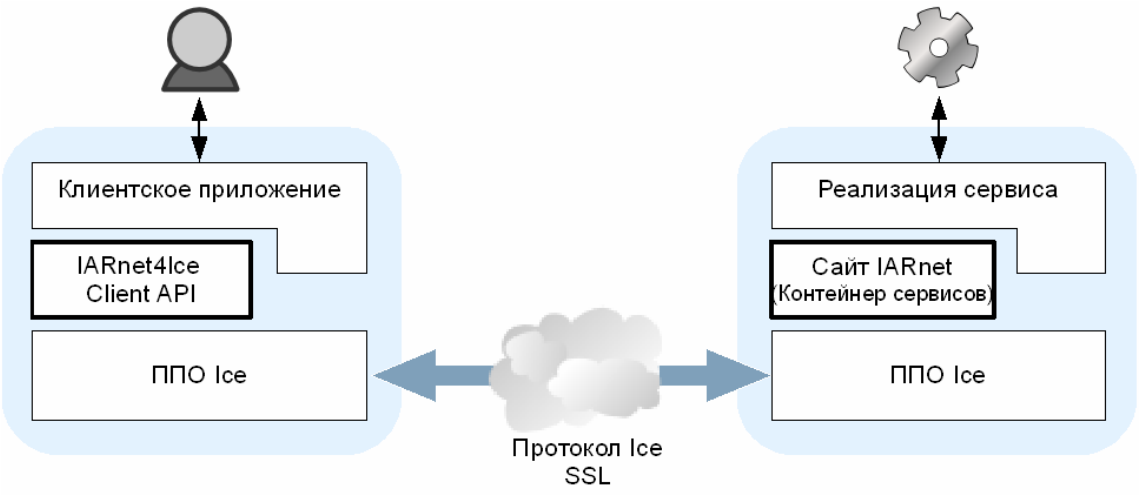

Рис.2. Базовые компоненты IARnet4Ice

версии IARnet, разработчик должен быть ознакомлен, по крайней мере, с основами программирования на Ice.

1.1.3. Интерфейсы сайта и сессии

В целях обеспечения безопасности, перед тем как начать использовать (вызывать) сервис, клиент должен пройти процедуру аутентификации на соответствующем сайте и создать *сессию*. Ниже приведено описание интерфейса сайта IARnet на языке Slice (Specification Language for Ice) (a).

Интерфейс *Site* имеет две операции для создания сессии, которые соответствуют различным способам аутентификации клиента: по имени и паролю или по цифровому сертификату, передаваемому по SSL-соединению. Это обеспечивает гибкость в выборе схемы аутен-

тификации клиента. Клиент также может осуществлять аутентификацию сайта. Для этого каждый сайт имеет цифровой сертификат, а взаимодействия между клиентом и сайтом всегда осуществляются по SSL-соединению.

После успешной аутентификации сайт возвращает клиенту *объект-сессию*, реализующий интерфейс для доступа к сервисам сайта. Ниже приведено описание интерфейса сессии на языке Slice (б).

Клиент может получить *прокси* (представитель Ice-объекта на клиентской стороне) требуемого сервиса с помощью операции *getService*, которой передается идентификатор сервиса. При обработке вызова данной операции сайт может провести дополнительную авторизацию клиента и отклонить запрос в случае, если данному клиенту не разрешен доступ к объекту. В случае успешной авторизации, клиенту возвращается прокси сервиса, с помощью которого клиент далее может взаимодействовать с сервисом как с обычным Ice-объектом.

Сессии имеют ограниченное время жизни. Клиент должен периодически продлевать время жизни сессии, вызывая операцию *keepAlive* на сессии. Сессия автоматически уничтожается сайтом в случае, если клиент не вызывал операцию *keepAlive* в течение определенного периода времени (таймаута), значение которого может быть получено с помощью операции *getSessionTimeout* из интерфейса сайта. При уничтожении сессии происходит освобождение всех используемых клиентом ресурсов. Данный механизм позволяет эффективно обрабатывать на стороне сайта отказы клиентов. Клиент также может явно уничтожить сессию, вызвав операцию *destroy*.

1.1.4. Архитектура сайта

Для обеспечения масштабируемости архитектура сайта IARnet разделена на две «распределяемые» части – *главный сервер* и *внутренний сервер* (Рис.3).

Главный сервер (front-end server) является точкой входа для клиентов. Он реализует аутентификацию и авторизацию клиентов, а также управление сессиями. Главный сервер также хранит информацию обо всех сервисах, размещенных на сайте. Главный сервер состоит из следующих компонентов:

- *Site Frontend* – реализация интерфейса *Site*;
- *Permissions Verifiers* – подключаемые модули аутентификации и авторизации, реализованные в виде Ice-объектов;
- *Session Manager* – менеджер сессий, управляющий созданием и уничтожением сессий;
- *Service Registry* – реестр сервисов, который хранит информацию о сервисах, размещенных на сайте, включая метаданные сервисов и адреса внутренних серверов.

Внутренний сервер (backend server) предназначен исключительно для размещения сервисов IARnet. Он реализует контейнер сервисов, который регистрирует размещенные в нем сервисы в реестре сервисов. После того, как клиент получил в рамках некоторой сессии прокси сервиса, все вызовы сервиса направляются прямо на соответствующий внутренний сервер, минуя главный

а)

```
interface Site {
   Session* createSession(string userId, string password)
      throws PermissionDeniedException, CannotCreateSessionException;
   Session* createSessionFromSecureConnection()
      throws PermissionDeniedException, CannotCreateSessionException;
   idempotent long getSessionTimeout();
};
```

б)

```
interface Session {
   Object* getService(string serviceId)
      throws ServiceNotFoundException, PermissionDeniedException;
   idempotent void keepAlive();
   void destroy();
};
```

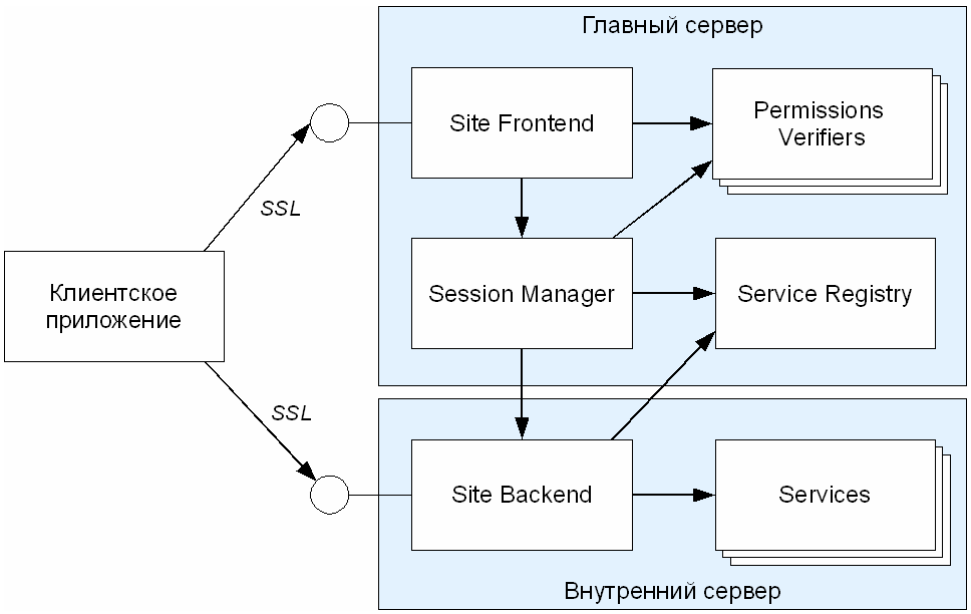

Рис.3. Архитектура сайта IARnet

сервер. Внутренний сервер производит авторизацию клиента на основе информации, полученной от главного сервера. А именно, главный сервер уведомляет внутренний сервер о сессиях, которым разрешен доступ к определенному сервису, а также – об уничтожении сессий.

Интерфейсы *ServiceRegistry* и *BackendServer* описывают контракт между главным и внутренним серверами. Это обеспечивает интероперабельность различных реализаций серверов. Например, сайт может включать внутренние серверы на различных языках программирования, скажем Java и C++.

Описанная архитектура поддерживает два варианта развертывания сайта – автономный и распределенный (Рис.4).

В *автономном режиме* сайт размещается полностью на одной машине и включает один внутренний сервер. Главный и внутренний серверы взаимодействуют друг с другом с помощью оптимизированных локальных Ice-вызовов, которые по скорости почти не уступают обычным локальным вызовам. В *распределенном режиме* сайт размещается на нескольких машинах и может включать несколько внутренних серверов. В этом случае, серверы сайта могут запускаться и управляться с помощью входящего в состав Ice сервиса IceGrid.

1.2. Текущая реализация

Описанная архитектура IARnet4Ice была реализована в виде прототипа на языке Java.

Реализация сайта основана на сервере IceBox, который предоставляет хорошую основу для контейнера сервисов IARnet, включая поддержку конфигурационных файлов, динамической загрузки сервисов и удаленного администрирования. Компоненты сайта загружаются с помощью специально реализованного IceBox-сервиса, который осуществляет инициализацию сервера в соответствии с его конфигурацией и режимом (автономный сайт, главный или внутренний сервер). Каждый сервис загружается с помощью другого специально реализованного IceBox-сервиса, который осуществляет инициализацию и регистрирует сервис на сайте.

Было реализовано два модуля аутентификации:

- MD5PermissionsVerifier, который реализует аутентификацию клиентов по имени и паролю с использованием файла с MD5-хэшами паролей;
- SimpleSSLPermissionsVerifier, который реализует аутентификацию клиентов по SSL-сертификату с использованием файла с именами доверяемых сертификатов.

Существующие сервисы Ice не реализуют механизм управления сессиями, который бы

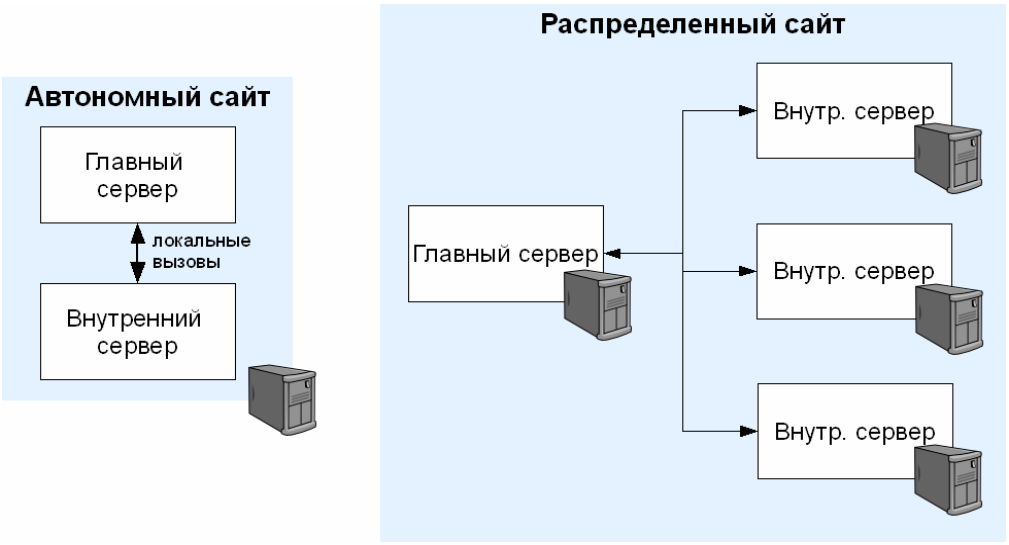

Рис.4. Варианты развертывания сайта

удовлетворял всем требованиям IARnet. Сервис Glacier2 содержит развитые средства аутентификации и управления сессиями, но обладает ограниченной применимостью и слишком неэффективен для использования в автономном режиме. IceGrid поддерживает аутентификацию клиентов и сессии только при доступе к резервируемым объектам. Поэтому входящий в состав главного сервера менеджер сессий был реализован полностью с нуля. В дальнейшем планируется исследовать вопрос об использовании Glacier2 для управления сессиями в распределенном режиме.

Клиенты IARnet используют контекст Ice-вызова для передачи идентификатора текущей сессии вместе с каждым вызовом сервиса. Внутренний сервер реализован с использованием механизма «servant locator», который позволяет выполнять любые действия во время диспетчеризации входящего вызова. В данном случае, реализация внутреннего сервера извлекает из контекста вызова идентификатор сессии и проверяет, имеет ли данная сессия доступ к вызываемому сервису. Если да, то вызов передается реализации сервиса. Если нет, клиенту возвращается исключение.

Модель программирования сервисов IARnet практически не отличается от разработки обычного Ice-объекта. Разработчик описывает интерфейс сервиса на языке Slice и компилирует Slice-описания в Java-код. После чего разработчик размещает сервис на сайте путем редактирования конфигурационного файла сервера и копирования jar-файла с реализацией серванта во внутреннюю директорию сервера. В отличие от обычного сервера IceBox, разработчику сервиса IARnet не требуется реализовывать код IceBox-сервиса с операциями *start* и *stop*. Данная функциональность обеспечивается стандартной реализацией загрузчика сервисов IARnet, который поддерживает передачу произвольных конфигурационных параметров конструктору серванта. Сервер IARnet4Ice также поддерживает обратные вызовы серванта при уничтожении сессии и остановке сервиса. Это позволяет реализовать управление внутренним состоянием сервиса и освобождение используемых сервисом ресурсов.

Распределенный режим развертывания сайта поддерживает управление инфраструктурой сайта с помощью сервиса IceGrid. Главные и внутренние сервера, включая размещенные на них сервисы, могут запускаться и управляться как IceGrid-приложения через удобный графический интерфейс администратора.

Интерфейс прикладного программирования IARnet4Ice Client API позволяет упростить управление сессиями и получение прокси-сервисов на клиентской стороне. Данный API реализует рутинные операции по созданию сессии, продлению времени жизни сессии, получению прокси сервисов и добавлению идентификатора сессии в контекст вызовов. После

получения прокси-сервиса, разработчик использует стандартную модель программирования Ice для взаимодействия с сервисом.

В текущей реализации IARnet4Ice также была реализована обратная совместимость с оригинальной версией IARnet. Это означает, что агенты доступа IARnet могут быть размещены на сайте IARnet и вызваны при помощи уже существующих клиентов. Данная функциональность обеспечивается специальной реализацией сетевого уровня IARnet.

2. Реализация сервиса обработки геоданных на основе IARnet4Ice

Одной из задач, поставленных в рамках проекта "Электронная Земля" [10], является организация удаленного доступа пользователей к существующим средствам обработки геоданных и соответствующей вычислительной инфраструктуре Grid.

Как показывает практика, наиболее удобным способом взаимодействия пользователя с Grid является использование проблемно-ориентированных Web-интерфейсов, т.н. *Grid-порталов*, скрывающих от пользователя детали взаимодействия с инфраструктурой Grid. Схема взаимодействия портала с Grid-системой основана на фиксации задачи на стороне Grid и использовании минимального числа параметризационных настроек, определяемых пользователем через Web-интерфейс портала.

Как правило, портал самостоятельно формирует низкоуровневые Grid-задания, находит подходящие ресурсы, осуществляет запуск заданий и контроль их выполнения. В настоящей работе был реализован другой, более гибкий подход, заключающийся в создании отдельного высокоуровневого сервиса обработки геоданных, выполняющего роль "брокера" между порталом и Grid. Данный сервис был реализован с использованием описанной выше экспериментальной версии IARnet4Ice.

2.1. Сервис GDPS

Сервис *GDPS (Geo Data Processing Service)* представляет собой универсальный сервис, предоставляющий доступ к различным прикладным и фундаментальным вычислительно сложным алгоритмам в области наук о Земле. Каждый алгоритм, обслуживаемый сервисом GDPS, имеет уникальный идентификатор, а также и набор входных и выходных параметров. Описания алгоритмов размещаются в распределенной базе метаданных проекта "Электронная Земля" как специальный тип ресурсов.

Взаимодействие пользователей с сервисом GDPS реализовано через центральный портал проекта "Электронная Земля" (Рис.5.). Пользователь формирует через Web-интерфейс портала набор данных, которые он хочет обработать. Затем пользователь выбирает из списка алгоритмов, представляемых сервисом, интересующий его алгоритм обработки данных и указывает параметры нового задания, в том числе – отобранные им данные. Далее пользователь инициирует запуск задания. Портал отправляет задание сервису, который производит загрузку данных, находит требуемый алгоритм и осуществляет его запуск локально или в рамках Grid. Пользователь может отслеживать статус своего задания через портал. После окончания выполнения задания, результаты задания передаются на портал и отображаются пользователю в одном из форматов геоданных, которые затем могут быть использованы при дальнейшем ГИС анализе.

2.2. Реализация сервиса

Рассмотрим основные принципы, которые использовались при реализации сервиса GDPS.

Исходные данные и результаты их обработки представляются в форме списка пар строк {<уникальное имя (ключ) параметра>, <значение параметра>}. Если речь идет о большом массиве данных, которые удобно передавать в виде файла, то в качестве значения параметра будет использоваться URL некоторого файла. При этом предполагается, что этот файл доступен удаленно по одному из общепринятых протоколов, таких как HTTP или FTP.

Сам сервис обработки данных реализует параллельное выполнение нескольких заданий, с возможностью ограничивать:

- число одновременно выполняемых заданий;
- общее число заданий, "хранящихся" сервисом, включая ожидающие, выполняемые и завершенные (результат которых еще не отправлен).

Приведем список используемых терминов:

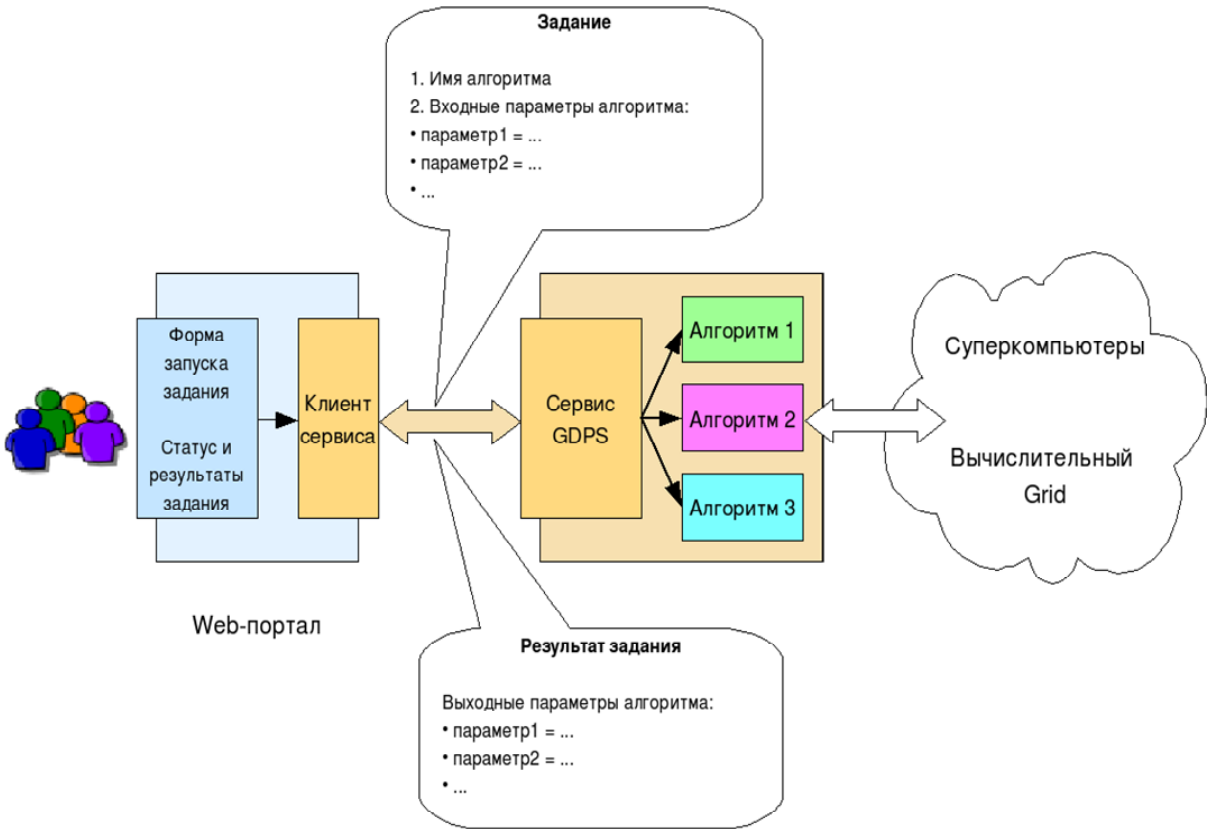

Рис.5. Схема взаимодействия портала и сервиса GDPS

- задача (*problem*) – описание задания, включая исходные данные и заказываемый алгоритм обработки (*solver*);

- задание (*job*) - процесс выполнения задачи на сервисе (другими словами, если задача принимается сервисом, то она «превращается» в задание);

- статус задания в очереди заданий - строки {"WAITING" | "PROCESSING" | "DONE"} (WAITING – означает, что задание ожидает выполнения, PROCESSING – задание выполняется, DONE – выполнение задания закончено, но результат еще не отправлен пользователю)

В свою очередь, каждое задание характеризуется:

- уникальным идентификатором, назначаемым процессором;

- временем формирования задания;

- статусом задания (см. выше).

Сервис формирует две очереди заданий: выполняемых и ожидающих выполнения. По завершению работы, задание сохраняется на сервисе до тех пор, пока клиентское приложение не запросит результат работы задания. Результат выполнения задания является массивом пар строк <имя параметра>:<значение>. В случае если выполнение задания завершилось ошибкой, результатом выполнения задания является краткое описание возникшей ошибки. Когда результат отправляется клиентскому приложению, задание удаляется из очереди сервиса.

Общая схема реализации и функционирования сервиса изображена на Рис.6.

Сервис GDPS реализован в виде сервиса IARnet с использованием экспериментальной версии IARnet4Ice. Интерфейс сервиса описан на языке Slice, реализация сервиса выполнена на языке Java и размещена на сайте IARnet. Использование IARnet4Ice в сравнении с оригинальной версией IARnet обладает следующими преимуществами:

- обеспечение безопасного взаимодействия между порталом и сервисом на основе протокола SSL и цифровых сертификатов;

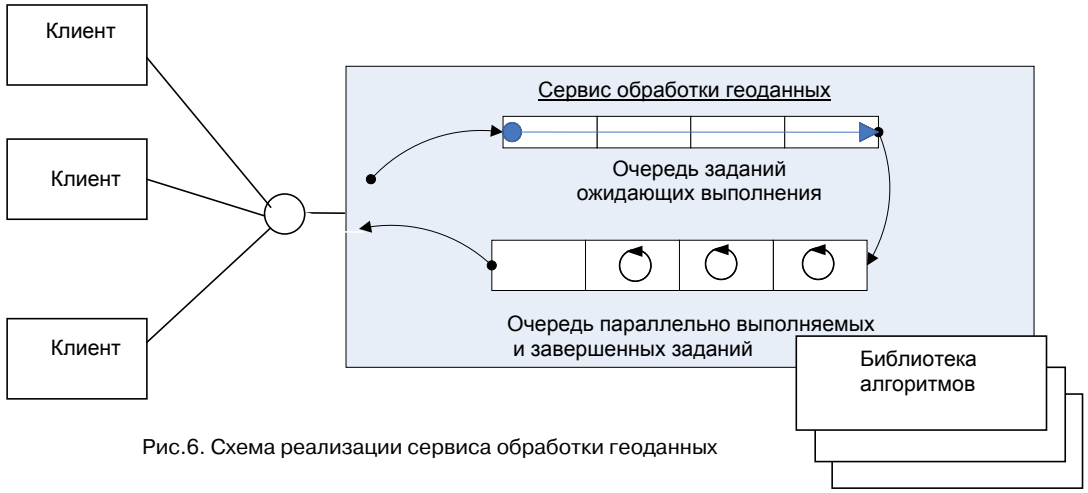

Рис.6. Схема реализации сервиса обработки геоданных

- описание довольно сложного интерфейса сервиса на языке Slice с определением новых типов;

- использование конфигурационных параметров для настройки сервиса;

- возможность использования ПО IceGrid для удаленного администрирования сервиса.

Сервис GDPS является универсальным в том смысле, что он поддерживает подключение произвольных реализаций алгоритмов обработки данных. Каждый подключаемый к сервису алгоритм должен расширять абстрактный Java-класс *AbstractSolver*. Подключение алгоритмов к сервису осуществляется через конфигурационные параметры сервиса.

В текущей реализации запуск алгоритмов осуществляется локально на сервере, предоставляющем сервис GDPS. В дальнейшем планируется реализовать поддержку выполнения вычислений на суперкомпьютере или в Grid путем запуска соответствующих низкоуровневых заданий.

Портал взаимодействует с сервисом GDPS при помощи клиентского приложения, реализованного на языке C++. Данное приложение предоставляет интерфейс командной строки для запуска, отмены, получения статуса и результатов заданий на удаленном сервисе GDPS.

2.3. Интеграция алгоритма обработки данных

В качестве демонстрационного примера на сервисе GDPS был размещен алгоритм RTL, разработанный в ИППИ РАН. Данный алгоритм реализует вычисление сеточного слоя, представляющего пространственно-временную характеристику сейсмического процесса

$$RTL(X,Y,T) = \sum_n \left[(E_n)^{\alpha} * \exp\left(-\frac{r_n}{R}\right) * \exp\left(-\frac{t_n}{T}\right) \right] * 1\left(\varepsilon - \frac{r_n}{R}\right) * 1\left(\varepsilon - \frac{t_n}{T}\right),$$

где: $1(t) = \begin{bmatrix} 1, t \geq 0 \\ 0, t < 0 \end{bmatrix}$, X,Y,T - координаты точки 3D-сетки, n – номер события, E – энергия, α - степень (например, 0.33), ε – пороговый коэффициент, r_n - расстояние от узла сетки до события, t_n – временной интервал от узла сетки до события, R,T - коэффициенты затухания.

Алгоритм RTL реализован на языке Java, поэтому легко может быть интегрирован в сервис GDPS путем разработки соответствующего адаптера.

Каждый алгоритм, установленный на сервисе GDPS, должен быть зарегистрирован в распределенной метабазе ресурсов проекта «Электронная Земля». Только после этого он станет доступным для поиска и использования через портал.

Метаописание алгоритма состоит из двух частей:

1. общего описания ресурса в соответствии со стандартной схемой метаданных,

2. специфичного для каждого алгоритма описания решаемой задачи, в виде спецификации входных и выходных параметров алгоритма.

Ниже приведены описание ресурса, соответствующего алгоритму RTL (в) и описание задачи, соответствующей алгоритму RTL (г).

Внутри элемента WWW размещается ссылка на описание задачи. Задача описывается в терминах входных (input) и выходных (output) параметров алгоритма. Для каждого параметра указывается его имя (name), тип (type) и расширенное название (title). Для входных парметров также могут быть указаны флаг обязательности (mandatory) и значение по умолчанию (value).

Адаптированный для использования в рамках сервиса GDPS алгоритм RTL имеет следующие входные параметры:

shape_file (shp) – shp-файл с каталогом событий (в виде URL);

R (double) – коэффициент затухания, в км. (см. формулу выше);

T (double) – коэффициент затухания, в сутках (см. формулу выше);

epsilon (double) – пороговый коэффициент (см. формулу выше);

alpha (double) – степень (см. формулу выше);

countX (int) – размер выходного растра по горизонтали;

countY (int) – размер выходного растра по горизонтали;

countT (int) – размер выходного растра по времени.

В качестве результата алгоритм RTL создает несколько файлов с вычисленным сеточным слоем, которые упаковываются в один zip-файл. Данный файл размещается на встроенном в сайт IARnet Web-сервере, а URL файла передается как выходной параметр алгоритма (`results_file`).

Внутри описания задачи также указывается физический адрес сервиса GDPS, на котором размещен данный алгоритм (атрибут location).

Информация, содержащаяся внутри описания задачи, используется порталом для динамического формирования Web-интерфейса пользователя и запуска задачи на сервисе. На Рис.7 изображен Web-интерфейс для запуска алгоритма RTL на сервисе GDPS, автоматически сгенерированный центральным порталом проекта «Электронная Земля» по описанию задачи.

После запуска задания пользователь может проверять статус выполнения задания на специальной странице. После выполнения задания на данной странице отображается его результат (Рис.8). В данном случае это ссылка на файл с результатами выполнения алгоритма RTL.

в)

```
<Database rno="2" section="e-Земля" type="27">
   <SourceRecord>ISA</SourceRecord>
   <Title>RTL</Title>
       <Description>
   Вычисление сеточного слоя, представляющего пространственно-временную характери-
стику сейсмического процесса.
       </Description>
   <CreatorOrg>ИППИ, ИСА</CreatorOrg>
   <WWW>http://dcs.isa.ru/eearth/problems/rtl.xml</WWW>
</Database>
```

г)

```
<problem name="RTL" location="GeoDataProcessor@DCS.ISA.RU:ssl -h dcs.isa.ru -p
7070">
   <input name="shape_file" mandatory="true" title="Shape file" type="shp"
value="http://dcs.isa.ru/eearth/data/hist_deh.shp"/>
   <input name="R" mandatory="true" title="R" type="double" value="1000"/>
   <input name="T" mandatory="true" title="T" type="double" value="1000000"/>
   <input name="alpha" mandatory="true" title="alpha" type="double" value="0.1"/>
   <input name="epsilon" mandatory="true" title="epsilon" type="double" value="0"/>
   <input name="countX" mandatory="true" title="countX" type="int" value="30"/>
   <input name="countY" mandatory="true" title="countY" type="int" value="30"/>
   <input name="countT" mandatory="true" title="countT" type="int" value="4"/>
   <output name="results_file" title="Results" type="string"/>
</problem>
```

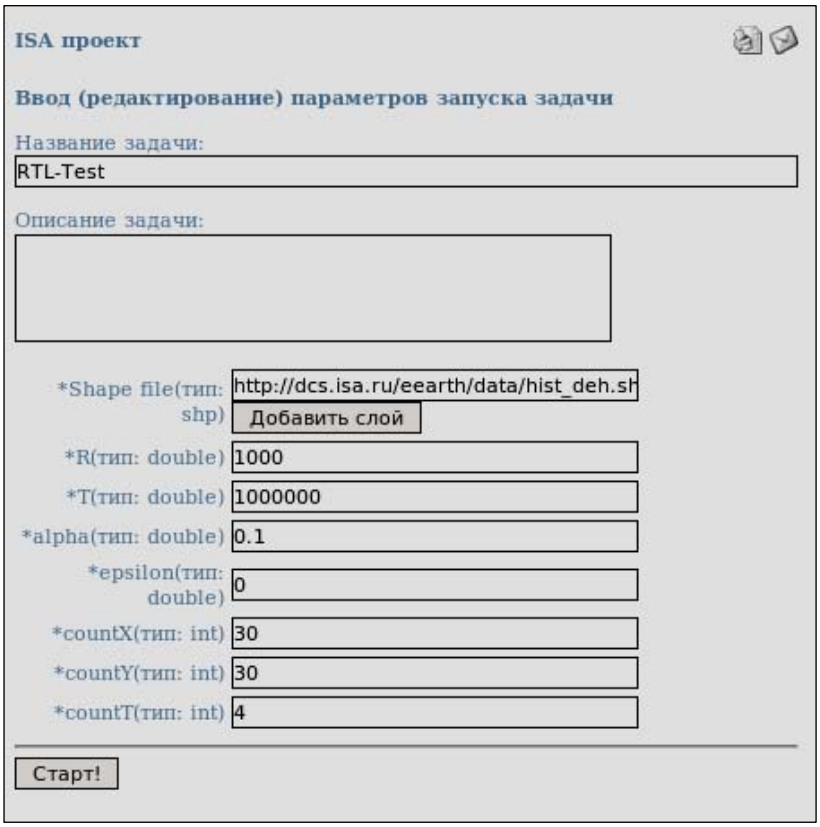

Рис.7. Web-интерфейс для запуска алгоритма

Рис.8. Результат выполнения задания

Заключение

В статье описаны архитектура и реализация экспериментальной версии инструментария IARnet, основанной на технологии Ice. Использование Ice позволило с минимальными усилиями добавить в инструментарий необходимую функциональность, такую как безопасность, управление сессиями, конфигурация и администрирование сервисов. Сервисы IARnet описываются с помощью полнофункционального и простого в использовании языка описаний интерфейсов. Сайт IARnet поддерживает модель Grid-сайта и различные режимы развертывания – от рабочей станции до нескольких серверов. Через подключаемые модули аутентификации IARnet4Ice поддерживает создание простых виртуальных организаций с центральным сервером аутентификации. Реализация обратной совместимости с оригинальной версией IARnet позволяет использовать совместно с новыми сервисами уже разработанные агенты доступа. Апробация экспериментальной версии IARnet была проведена на примере разработки высокоуровневого сервиса обработки геоданных в рамках проекта “Электронная Земля”.

Литература

1. Емельянов С.В., Афанасьев А.П., Волошинов В.В., Гринберг Я.Р., Кривцов В.Е., Сухорослов О.В. Реализация Grid-вычислений в среде IARnet. // Информационные технологии и вычислительные системы. - М.: 2005, №2, с.61-75.
2. Alexander Afanasiev, Oleg Sukhoroslov, Mikhail Posypkin. A High-Level Toolkit for Development of Distributed Scientific Applications. // Victor E. Malyshkin (Ed.): Parallel Computing Technologies, 9th International Conference, PaCT 2007, Pereslavl-Zalessky, Russia, September 3-7, 2007, Proceedings. Lecture Notes in Computer Science 4671 Springer 2007, ISBN 978-3-540-73939-5, pp. 103-110.
3. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001.
4. Foster, I. Service-Oriented Science. Science, vol. 308, May 6, 2005.
5. Foster, I. Scaling eScience Impact. 1st Iberian Cyberinfrastructure Conference, Santiago de Compostela, Galicia, Spain, May 15, 2007. http://www-fp.mcs.anl.gov/~foster/Talks/Scaling%20eScience%20IBERGrid.pdf
6. Thomas Erl. Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, 2005.
7. Henning, M.: A New Approach to Object-Oriented Middleware. IEEE Internet Computing, vol. 08, no. 1 (2004) 66-75
8. Michi Henning, Mark Spruiell. Distributed Programming with Ice. ZeroC, Inc. Revision 3.2.1, August 2007. http://www.zeroc.com/download/Ice/3.2/Ice-3.2.1.pdf
9. Сухорослов О.В. Промежуточное программное обеспечение Ice. // Проблемы вычислений в распределенной среде. Труды ИСА РАН, 2007 (в печати)
10. Центральный портал проекта “Электронная Земля” http://eearth.viniti.ru/

Афанасьев Александр Петрович. Заведующий Центром распределенных вычислительных систем ИСА РАН. Окончил МФТИ в 1968 году. Доктор физико-математических наук. Автор свыше 100 научных работ. Специалист в области оптимального управления, распределенных вычислений, информационно-телекоммуникационных систем.

Волошинов Владимир Владимирович. Старший научный сотрудник ИСА РАН. Окончил МФТИ в 1984 году. Кандидат физико-математи еских наук. Автор свыше 30 научных работ. Специалист в области прикладной математики, программного обеспечения, распределенных вычислений.

Сухорослов Олег Викторович. Научный сотрудник ИСА РАН. Окончил МФТИ в 2002 году. Кандидат физико-математических наук. Автор 10 научных работ. Специалист в области распределенных систем и Grid-вычислений.