Архитектура и принципы реализации системы IARnet

В. В. Волошинов, О. С. Естехин, О. В. Сухорослов

В работе рассматривается программная архитектура и принципы реализации системы IARnet, базирующейся на описанном в [1] подходе к построению распределенной вычислительной среды (РВС) [2]. Основу подхода составляет понятие информационно-алгоритмического ресурса (ИАР), как абстракции, позволяющей описать общую модель доступа к ресурсам РВС и использовать РВС для решения широкого класса прикладных задач.

1. Требования к системе

В данном разделе описываются требования к универсальной программной инфраструктуре распределенной вычислительной среды, соответствующей рассмотренной в [1] архитектуре.

Под **требованием** будем понимать некоторое качество, которым должна обладать разрабатываемая система для того, чтобы решать поставленную задачу. Требования к системе могут быть разделены на две группы: функциональные и нефункциональные. Функциональные требования описывают видимые пользователям возможности системы. Нефункциональные требования включают требования к удобству использования, требования к надежности, требования к производительности и т. п.

1.1. Функциональные требования

Общие требования

Система должна:

- реализовывать высокоуровневую модель доступа к ИАР, описанную в [1];
- поддерживать архитектуру распределенной вычислительной среды, описанную в [1];
- предоставлять средства:
  - — для разработки приложений (клиентская часть);
  - — для интеграции ресурсов в распределенную вычислительную среду (серверная часть);

  - для организации распределенной вычислительной среды (служебные сервисы);
- поддерживать работу в открытых глобальных сетях и обеспечивать взаимодействие пользователей и ресурсов, находящихся в локальных сетях различных организаций;
- предоставлять ресурсу (службе) средства для вызова других ресурсов и служб РВС;
- предоставлять механизмы оповещения об ошибках, диагностирования, журналирования и т. п.

Первичные ресурсы

Система должна поддерживать работу:

- с различными классами первичных ресурсов;
- с гетерогенными первичными ресурсам;
- с первичными ресурсами, функционирующими на различных аппаратных платформах и под управлением различных операционных систем;
- с первичными ресурсами, реализованными на различных языках программирования;
- с первичными ресурсами, принадлежащими и управляемыми различными организациями и лицами;
- с первичными ресурсами, находящимися за межсетевым экраном.

Коммуникационные механизмы

- Система должна позволять разработчикам агентов доступа использовать различные коммуникационные механизмы для организации доступа к ресурсам.

Интеграция первичных ресурсов

- Система должна предоставить :
  - спецификацию серверного API, описывающую модель интеграции первичных ресурсов в РВС на основе агентов доступа;
  - реализацию серверного API.
- Серверный API не должен зависеть от коммуникационного механизма доступа к ресурсу.
- Система должна поддерживать:
  - гибкий, легкий в использовании и простой механизм интеграции первичного ресурса (требующий от разработчикам агентов доступа создания минимального объема программного кода);
  - одновременную работу с ресурсом нескольких пользователей;
  - гибкий, легкий в использовании и простой механизм развертывания ресурса в сети на основе различных коммуникационных механизмов доступа.

- Система должна:
  - — скрывать от разработчиков агентов доступа детали связывания с конкретным коммуникационным механизмом доступа к ресурсу;
  - — позволять разработчикам агентов доступа самостоятельно реализовывать связывание с конкретным коммуникационным механизмом доступа к ресурсу;
  - — (по возможности) поддерживать развертывание агентов доступа на машинах с минимальными системными ресурсами.

Описание ресурсов

Система должна:

- поддерживать стандартный язык описания интерфейсов ресурсов, не зависящий от языков программирования;
- предоставлять стандартные средства для описания метаданных ресурсов.

Типизация ресурсов

Система должна:

- позволять определять типы ресурсов на основе описания стандартного интерфейса ресурсов данного типа;
- поддерживать унифицированный доступ к ресурсам одного типа. Все ресурсы определенного типа должны быть доступны через одинаковый интерфейс.

Служебные сервисы

Система должна поддерживать:

- динамический механизм обнаружения ресурсов и служб;
- механизм мониторинга состояния ресурсов, служб, узлов и сетевых соединений;
- механизм обмена сообщениями между ресурсами на основе модели уведомлений (publish-subscribe);
- механизм ведения журналов работы распределенных приложений;
- описание и автоматическое выполнение прикладных сценариев координированного использования нескольких ресурсов.

Модель прикладного программирования

- Система должна предоставить спецификацию клиентского API, описывающую модель доступа к ресурсам и службам РВС.
- Для доступа к службам должна использоваться та же модель, что и для доступа к ресурсам.
- Система должна:
  - — предоставить реализацию клиентского API;

  - позволять приложениям использовать ресурсы и службы путем вызова их операций через клиентский API;
  - скрывать от приложения детали реализации доступа к ресурсам и службам.
- Прикладной программист не обязан знать:
  - каким образом обнаруживаются ресурсы (прозрачность поиска);
  - где физически находятся используемые им ресурсы (прозрачность местонахождения). Метод вызова ресурса (службы) не должен зависеть от его местонахождения;
  - каким образом реализуется удаленный доступ к ресурсу (прозрачность коммуникационного механизма). Метод вызова ресурса (службы) не должен зависеть от используемого коммуникационного механизма.
- Метод вызова ресурса (службы) не должен зависеть от деталей его реализации и развертывания.
- Система должна:
  - поддерживать синхронные, отложенные синхронные и асинхронные вызовы ресурсов и служб;
  - позволять приложению динамически связываться с конкретными ресурсами и службами во время его выполнения;
  - позволять приложению динамически обнаруживать ресурсы и службы во время его выполнения;
  - предоставлять средства для получения приложением всей требуемой информации о ресурсе;
  - (по возможности) поддерживать запуск приложений на машинах с минимальными системными ресурсами.
- Система должна позволять прикладному программисту:
  - контролировать процесс доступа к ресурсам и службам;
  - легко создавать приложения путем модификации унаследованного кода;
  - работать максимально независимо от текущей конфигурации РВС (набор доступных ресурсов, служб и т. д.).

Безопасность

- Механизм безопасности должен быть максимально прозрачен для приложений и пользователей приложений.
- Система должна поддерживать:
  - глобальный механизм обеспечения безопасности;
  - аутентификацию пользователей и ресурсов;
  - описание политики доступа к ресурсу и авторизацию пользователей;

  - механизм ведения журнала доступа к ресурсу.
- Система должна:
  - позволять администратору ресурса определять и управлять политикой доступа к ресурсу;
  - использовать проверенные стандарты, механизмы безопасности и существующие реализации;
  - поддерживать современную инфраструктуру безопасности Grid Security Infrastructure (GSI).

Реализации

- Система должна иметь реализации API на основных языках программирования (как минимум — Java на клиентской и серверной стороне, желательно также C++ на серверной).

1.2. Нефункциональные требования

- Архитектура системы должна быть открытой, т. е. построенной на четко описанных стандартах.
- Система должна:
  - поддерживать возможность создания сторонних реализаций стандартных API и их использования вместе с эталонными реализациями;
  - поддерживать интеграцию с современными технологиями ППО и Grid-технологиями;
  - быть расширяемой и учитывать возможность внесения изменений в будущем;
  - быть максимально переносимой, т. е. поддерживать запуск на различных аппаратных и программных платформах;
  - быть масштабируемой, т. е. добавление новых пользователей и новых ресурсов не должно заметно снижать эффективность доступа к существующим ресурсам;
  - быть отказоустойчивой, т. е. устойчивой по отношению к отказам/выходам из системы отдельных ресурсов, служб, узлов и сетевых соединений;
  - предоставлять гибкие, легкие в использовании и простые API;
  - иметь информативные и прозрачные интерфейсы с ясными статусными сообщениями и сообщениями об ошибках;
  - включать хорошо документированный программный код, руководства пользователей и примеры использования;
  - позволять разработчику работать максимально независимо от текущей конфигурации РВС (набор доступных ресурсов, служб и т. д.);
  - поддерживать быстрое создание прототипов и тестирование приложений, ресурсов и других компонентов;

— поддерживать легкое развертывание и установку;
— быть легкой в сборке;
— предоставлять набор тестов для проверки корректного функционирования системы.

1.3. Проектные ограничения

- Спецификации клиентской, серверной и служебных частей должны проводиться на основе описания стандартных API.
- При разработке системы должны максимально широко использоваться существующие стандарты.
- Для решения типовых задач система должна максимально широко использовать существующие технологии и реализации.

2. Программная архитектура системы IARnet

В рамках данной работы была разработана система IARnet [3], которая реализует универсальную программную инфраструктуру распределенной вычислительной среды, удовлетворяющую описанным выше требованиям.

На рис. 1 представлена архитектура IARnet. Уровню агентов доступа соответствуют *агенты доступа* и *контейнеры ИАР*. Уровень прикладного API образует *клиентская библиотека*, которая использует для вызова ресурсов так называемые *коннекторы*. Службы IARnet реализованы в виде ИАР-ов, в соответствии с приведенными в [1] соображениями. Для вызова других компонентов РВС ресурсы и службы используют те же средства, что и приложения, т. е. клиентскую библиотеку и набор коннекторов.

Рассмотрим основные элементы данной архитектуры.

2.1. Агент доступа

Агент доступа в IARnet представляет собой программный компонент, реализующий набор операций, через которые осуществляется доступ к функциональности первичного ресурса. При вызове подобной операции агент доступа может делегировать ее выполнение первичному ресурсу путем вызова данного ресурса через его программный интерфейс. Агент также может сам произвести все требуемые действия по выполнению операции, являясь при этом полноценным ресурсом. В любом случае, детали выполнения операций, предоставляемых агентом, скрыты от остальных элементов системы.

В общем случае, агент доступа может не находиться на той же машине, что и первичный ресурс. При этом вместо локальных вызовов агент доступа использует определенный механизм удаленного доступа к первичному ресурсу. Выбор места размещения агента доступа и способа вызова первичного ресурса не ограничены архитектурой и определяются разработчиком агента доступа индивидуально в каждом конкретном случае.

Помимо задачи унификации интерфейса первичного ресурса, которая в общем случае является уникальной для каждого ресурса, агент

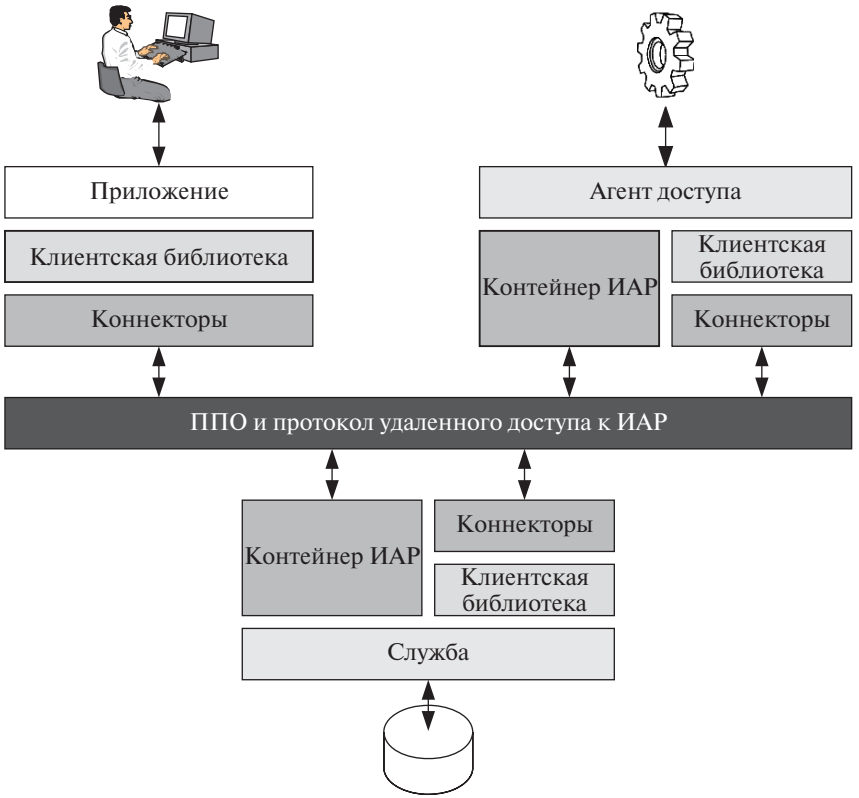

Рис. 1. Архитектура IARnet

доступа должен решать такие стандартные задачи, как контроль доступа к ресурсу, обеспечение одновременной работы с ресурсом нескольких пользователей, учет использования ресурса и т. д. Для этих задач в IARnet предусмотрены стандартные реализации, которые разработчик агента доступа может использовать при создании агента и сосредоточиться на унификации функциональности ресурса. Разработчик также может переопределить стандартные реализации данных механизмов.

В соответствии с вышенаписанным, интерфейс агента доступа состоит из двух основных частей (рис. 2): стандартного общего интерфейса, который должен реализовать каждый агент доступа, и интерфейса доступа к соответствующему типу ресурса. Для первого интерфейса предусмотрены стандартные реализации, а реализация второго осуществляется разработчиком агента доступа индивидуального для каждого конкретного первичного ресурса.

Агент доступа в IARnet не решает задачу организации удаленного доступа к ресурсу. Поскольку данная задача может быть решена при помощи различных технологий ППО и коммуникационных механизмов, то для обеспечения открытости и расширяемости архитектуры организация удаленного доступа изолирована в другом компоненте — контейнере ИАР.

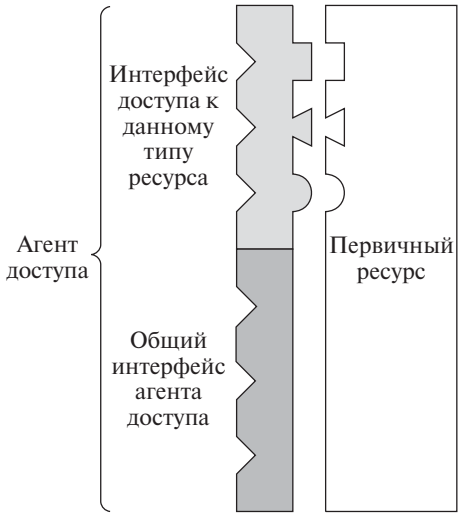

Рис. 2. Внешние интерфейсы агента доступа

2.2. Контейнер ИАР

Контейнер ИАР представляет собой среду, в которой развертываются и функционируют агенты доступа. Контейнер решает две задачи:

- обеспечивает удаленный доступ к агентам посредством определенного коммуникационного механизма и технологии ППО;
- предоставляет унифицированный интерфейс для развертывания агентов доступа и их администрирования.

Архитектура IARnet поддерживает наличие различных реализаций контейнеров ИАР. Схема устройства контейнера ИАР приведена на рис. 3.

Для решения первой задачи конкретная реализация контейнера ИАР использует определенную реализацию ППО (заштрихованная область на схеме), в которой размещаются представители агентов доступа (темные квадраты на схеме), доступные удаленно посредством соответствующего протокола ППО.

Например, в случае использования технологии CORBA, реализация ППО соответствует определенной реализации брокера объектных запросов ORB, представители агентов доступа — CORBA-объектам, а протокол ППО — протоколу IIOP. В случае использования технологии Web-сервисов, реализация ППО соответствует определенной реализации среды размещения Web-сервисов (например, Apache Axis), представители агентов доступа — Web-сервисам, а протокол ППО — протоколу SOAP.

Процедура развертывания агента доступа в контейнере заключается в создании и развертывании в слое ППО представителей агентов доступа. На входе данной процедуры разработчик предоставляет контейнеру готовый агент доступа вместе с его настройками, а на выходе он получает *ссылку* на развернутый в среде ИАР. Данная ссылка затем может быть использована на клиентской стороне для вызова ресурса посредством

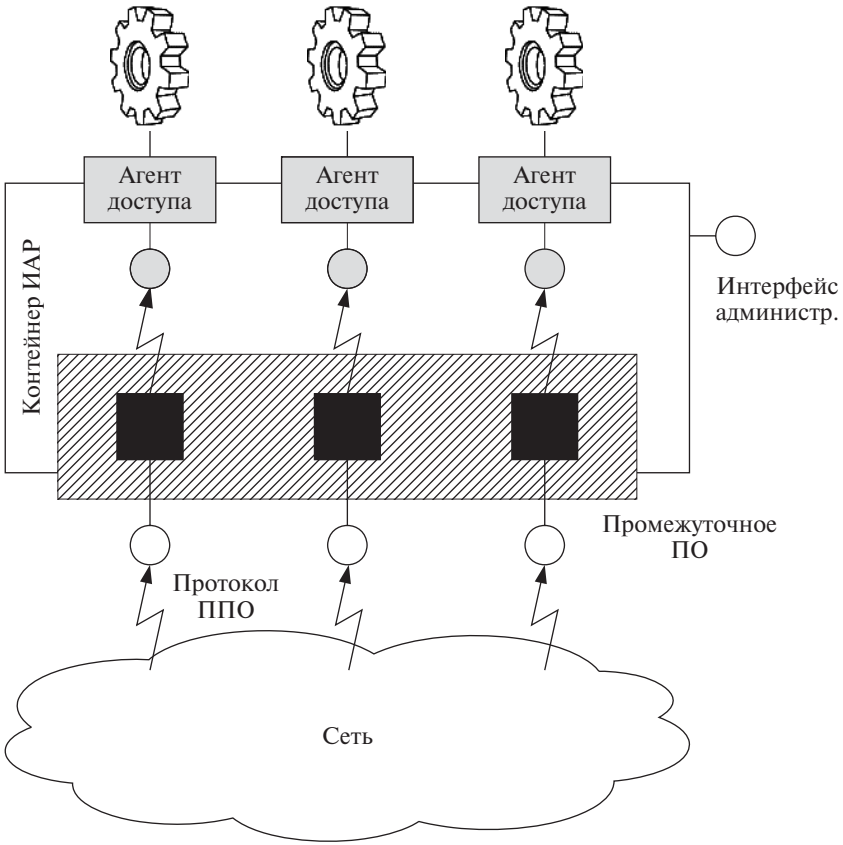

Рис. 3. Контейнер ИАР

соответствующего коннектора и протокола ППО. Приходящие вызовы передаются средствами реализации ППО представителям агентов доступа, которые делегируют обработку вызовов агентам.

После развертывания агента доступа может быть проведена регистрация ИАР в среде на основе полученной при развертывании ссылки и описания ресурса, предоставленного его владельцем.

Развертывание агентов доступа и удаление их из контейнера, а также администрирование агентов (управление доступом к ресурсу) осуществляется через унифицированный интерфейс контейнера ИАР. Данный интерфейс определяет контракт на разработку контейнера, т. е. разработчик конкретного контейнера ИАР должен реализовать этот интерфейс. При этом детали реализации контейнера скрыты как от разработчика агента доступа, так и от администратора ресурса.

2.3. Клиентская библиотека

Клиентская библиотека IARnet представляет собой реализацию интерфейса прикладного программирования IARnet Client API, который описывает высокоуровневую модель доступа к ИАР. IARnet Client API включает в себя:

- Интерфейс *представителя ресурса* — объекта на клиентской стороне, который разработчик приложения использует для взаимодействия с ресурсом.
- Интерфейс для получения представителя ресурса по ссылке на ИАР.
- Клиентские интерфейсы к службам IARnet.

2.4. Коннекторы

Для прозрачной поддержки различных технологий и протоколов ППО для доступа к ресурсам, в архитектуре IARnet на клиентской стороне был введен отдельный уровень коннекторов (см. рис. 4).

Коннектор представляет собой программный компонент на клиентской стороне, скрывающий за стандартным интерфейсом детали реализации доступа к ресурсу посредством определенного коммуникационного механизма. Стандартный интерфейс коннектора, определенный в архитектуре IARnet, описывает контракт на разработку коннектора.

При создании представителя ресурса на клиентской стороне происходит автоматическое связывание его с соответствующим коннектором, который поддерживает требуемый для доступа к ресурсу коммуникационный механизм. Сопоставление коннектора ресурсу на клиентской стороне производится на основе информации, содержащейся в ссылке на ИАР, — так называемого профиля.

Напомним, что ссылка на ИАР генерируется контейнером ИАР во время развертывания агента доступа. В зависимости от того, какой механизм (или механизмы) удаленного доступа поддерживает контейнер, в ссылку вставляется информация, позволяющая на клиентской стороне однозначно определить, какой коннектор использовать для работы с ресурсом. В общем случае контейнер ИАР может поддерживать несколько механизмов удаленного доступа, при это для каждого из данных механизмов должен быть создан и общедоступен соответствующий коннектор.

Таким образом, коннекторы и контейнеры ИАР образуют слой, скрывающий от вышележащих слоев архитектуры различные коммуникационные механизмы, технологии ППО и протоколы. Точнее, коммуникационный механизм скрыт внутри «черного ящика», клиентскую часть которого образует коннектор, а серверную — представитель агента доступа в слое ППО, используемого контейнером. Коннектор принимает от представителя ресурса параметры вызова ресурса, подготавливает их к пересылке и производит вызов представителя агента доступа. Представитель агента доступа принимает вызов от коннектора, производит перевод параметров вызова в стандартное представление и делегирует выполнение вызова

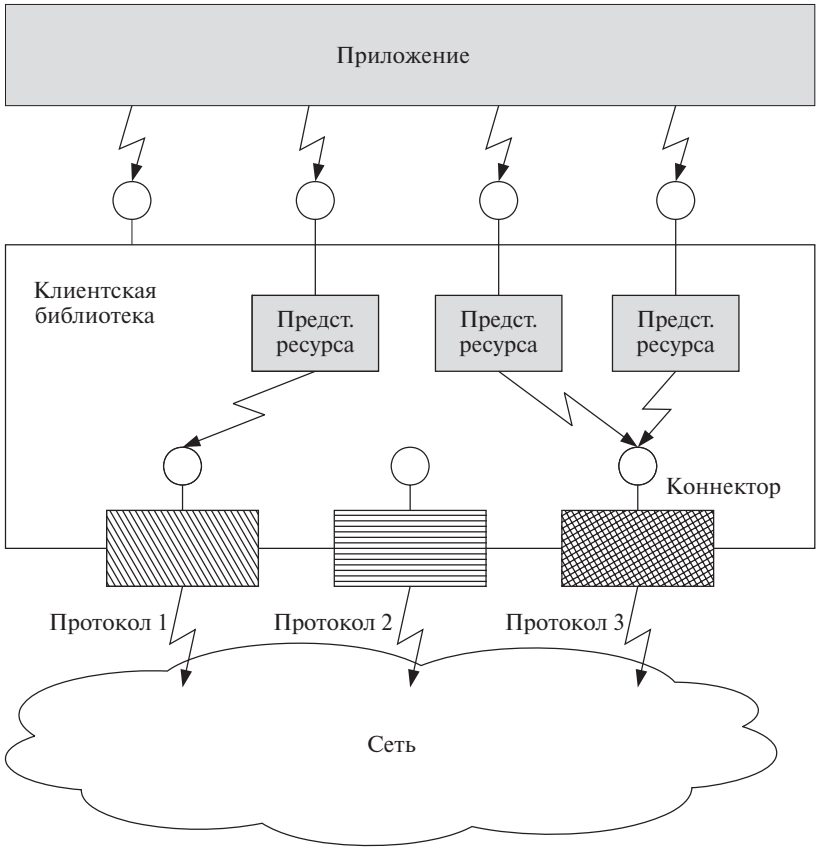

Рис. 4. Клиентская библиотека и коннекторы

агенту доступа. После выполнения вызова агентом доступа результат вызова отправляется представителем агента доступа обратно коннектору, который передает его представителю ресурса на клиентской стороне.

2.5. Службы

Для обеспечения общности и однородности архитектуры, службы IARnet реализуются в виде агентов доступа и развертываются в контейнерах ИАР. Каждый из подобных агентов доступа должен реализовывать унифицированный интерфейс одной из служб IARnet.

В случае если используется реализация службы из существующей Grid-инфраструктуры, агент доступа осуществляет интеграцию данной реализации в РВС в виде службы IARnet. В случае создания реализации службы IARnet, агент доступа непосредственно содержит реализацию данной службы.

Таким образом, для создания службы IARnet разработчику необходимо создать агент доступа, реализующий унифицированный интерфейс данной службы. Агент может содержать полную реализацию службы или делегировать выполнение вызовов службе Grid-инфраструктуры. Развертывание службы в среде осуществляется путем развертывания данного агента доступа в контейнере ИАР.

Соответственно, доступ к службе на клиентской стороне осуществляется в рамках той же модели, что и доступ к ресурсам. По ссылке на службу (ИАР) прикладной программист получает объект-представитель службы. Представитель службы может быть полностью аналогичен представителю ресурса или реализовывать более удобный клиентский интерфейс к службе.

В рамках IARnet были предусмотрены следующие базовые службы:

- Служба регистрации, осуществляющая регистрацию и поиск ресурсов и других служб РВС.
- Служба мониторинга, осуществляющая мониторинг состояния ресурсов и других компонентов среды.
- Служба рассылки сообщений, реализующая механизм подписки на уведомления о событиях, связанных с ресурсами.
- Служба журналов, позволяющая вести единый системный журнал распределенного приложения.
- Служба безопасности, осуществляющая выдачу цифровых сертификатов.

В рамках IARnet предусмотрена следующая высокоуровневая служба:

- Служба управления заданиями, осуществляющая запуск и управление ходом выполнения заданий пользователей.

Заключение

В данной статье были рассмотрены требования к универсальной программной инфраструктуре РВС и основные элементы программной архитектуры системы IARnet, реализующей подобную инфраструктуру. За рамками статьи остался целый ряд вопросов, относящихся как к архитектуре системы IARnet, так и к ее реализации:

- Реализация агента доступа.
- Идентификация ресурсов.
- Описание и поиск ресурсов.
- Реализации коннекторов и контейнеров.
- Описание служб IARnet и их реализаций.
- Обеспечение безопасности.

Данные вопросы планируется подробно рассмотреть в последующих статьях.

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.: Институт микропроцессорных вычислительных систем РАН, 2005. № 2.
2. *Афанасьев А. П.*, *Волошинов В. В.*, *Рогов С. В.*, *Сухорослов О. В.* Развитие концепции распределенных вычислительных сред // Проблемы вычислений в распределенной среде: организация вычислений в глобальных сетях. Сборник трудов ИСА РАН. М.: УРСС, 2004.
3. *Афанасьев А. П.*, *Волошинов В. В.*, *Кривцов В. Е.*, *Рогов С. В.*, *Сухорослов О. В.* Использование информационно-алгоритмических ресурсов для организации распределенных вычислений // Проблемы вычислений в распределенной среде: организация вычислений в глобальных сетях. Сборник трудов ИСА РАН. М.: УРСС, 2004.