Сервис-ориентированный подход к организации распределенных вычислений с помощью инструментального комплекса DISCENT

И.В. Бычков, Г.А. Опарин, А.Г. Феоктистов, В.Г. Богданова, А.С. Корсуков

Аннотация. В статье представлен подход к организации распределенных вычислительных сред на основе грид-сервисов. Сервис-ориентированные среды обладают рядом свойств, существенно усложняющих процессы их организации и применения. Анализ мировых тенденций в области автоматизации решения прикладных задач в подобных средах позволяет утверждать, что решение этих проблем непосредственно связано с интеллектуализацией сервис-ориентированного инструментария. В данной статье рассматривается пример такого инструментария – комплекс DISCENT, представляющий собой промежуточное программное обеспечение кластерной грид, в которой управление кластерами осуществляется с помощью стандартизированных систем управления заданиями. Особенность представленного подхода заключается в интеграции методов сервис-ориентированного программирования с оригинальными интеллектными методами и средствами управления распределенными вычислениями, разработанными в Институте динамики систем и теории управления (ИДСТУ) СО РАН на основе парадигм концептуального программирования, баз знаний и грид-вычислений.

Ключевые слова: распределенные вычисления, вычислительные задания, сервис-ориентированное программирование, конкретизация вычислительных заданий.

Введение

При организации распределенных вычислительных сред важной практически значимой проблемой является размещение в них проблемно-ориентированных программных комплексов и обеспечение удаленного высокоуровнего доступа пользователей-“предметников” к этим комплексам. В настоящее время решение данной проблемы осуществляется, как правило, на основе концепции Service-Oriented Science [1], базирующейся на автоматизации интеграции распределенных гетерогенных ресурсов в виртуальную исполнительную среду и прозрачного использования этой среды путем представления ресурсов в виде сервисов. Известно применение научных сервисов в вычислительной химии, компьютерной алгебре, статистическом анализе временных рядов (статистическом прогнозировании), биоинформатике, геоинформатике и во многих других областях. Сервис-ориентированные среды обладают рядом свойств, существенно усложняющих процессы их организации, исследования и применения. В их числе: организационно-функциональная разнородность и динамичность; разнообразие спектра решаемых задач; наличие различных категорий пользователей, преследующих свои субъективные цели эксплуатации тех или иных ресурсов; применение в узлах среды различных средств управления вычислениями. Анализ мировых тенденций в области автоматизации решения прикладных задач в таких средах позволяет утверждать, что решение этих проблем непосредственно связано с интеллектуализацией сервис-ориентированного инструментария.

К настоящему времени разработан широкий спектр инструментов для построения сервис-ориентированных распределенных вычисли-

тельных сред. К их числу относятся, например, средства общего назначения Amazon Elastic Compute Cloud, Google App Engine и Microsoft Windows Azure, специализированные системы Globus Toolkit, Unicore, языки программирования Opa и Swarm. Из отечественных средств построения сервис-ориентированных сред следует отметить такие системы, как MathCloud [2], iPSE [3] и сервис-ориентированная ГИС ННЦ СО РАН [4]. Основной упор в таких системах, как правило, делается на упрощение процесса создания сервисов. Однако существует ряд важных задач, связанных с организацией распределенных и параллельных вычислений в сервис-ориентированных средах, требующих проведения дополнительных исследований, в частности, создание и публикация новых сервисов с использованием соответствующих аппаратных и программных ресурсов СО РАН [5].

В статье представлен подход к оформлению приложений, ориентированных на работу в распределенной вычислительной среде в виде удаленно доступных сервисов с помощью средств инструментального комплекса DISCENT [6], разрабатываемого в Институте динамики систем и теории управления СО РАН (ИДСТУ СО РАН).

1. Приложения и сервисы

Веб-сервис [7] (или веб-служба) представляет собой идентифицируемую веб-адресом программную систему со стандартизированными интерфейсами. Эти интерфейсы обеспечивают обмен данными между различными приложениями, реализованными с помощью различных языков программирования и размещенными в разнородных узлах вычислительной среды. С появлением веб-сервисов развилась концепция сервис-ориентированной архитектуры веб-приложений (Service Oriented Architecture).

Сегодня наиболее популярные средства реализации веб-сервисов базируются на применении протокола Simple Object Access Protocol (SOAP) или стиля Representational State Transfer (REST). Протокол SOAP является развитием протокола вызова удаленных процедур XML Remote Procedure Call (XML-RPC). Технология организации веб-служб с помощью этого протокола включает [8, 9]: язык описания веб-сервисов Web Services Description Language (WSDL); стандарт универсального описания, обнаружения и интеграции Universal Description, Discovery and Integration (UDDI), предназначенный для определения, регистрации и обнаружения веб-служб; протокол SOAP, по которому происходит обмен сообщениями между удаленным приложением и клиентом. REST – это стиль построения архитектуры распределенного приложения. Стиль REST основывается на концепции манипуляции объектами с помощью четырех методов: “создать”, “прочесть”, “обновить”, “удалить”. Таким образом, REST является довольно простым и прозрачным подходом к организации веб-служб. Как правило, протокол SOAP применяют в сложных архитектурах, где взаимодействие с объектами выходит за рамки методов, предоставляемых стилем REST.

Яном Фостером и его коллегами была разработана архитектура Open Grid Service Architecture (OGSA), базирующаяся на сервисах нового типа – грид-сервисах [10]. Имеется ряд существенных отличий между грид-сервисами и веб-сервисами. Основным из них является то, что веб-сервисы функционируют независимо от клиентов. Что же касается грид-сервисов, то экземпляр определенного сервиса может быть специально создан в удаленном узле вычислительной среды по запросу от клиента. Созданный экземпляр грид-сервиса будет предоставлять вычислительные услуги клиенту, обратившемуся с запросом к грид-сервису. При предоставлении услуги клиенту доступны сервисные метаданные и сведения о состоянии сервиса, а также промежуточные и итоговые результаты вычислений. По завершению выполнения запроса клиента экземпляр грид-сервиса удаляется.

Представленный в статье подход ориентирован на создание и применение грид-сервисов с использованием протокола SOAP. Данный протокол был выбран, благодаря его надежности и безопасности, а также отсутствия каких-либо ограничений по сравнению со стилем REST при организации грид-сервисов и взаимодействия с ними.

Для того, чтобы создать грид-сервис на основе приложения, требуется сформировать описание сервиса на языке WSDL. WSDL-описание грид-сервиса должно содержать уни-

фицированный идентификатор ресурса, в котором размещен сервис, интерфейс взаимодействия клиента с сервисом, типы используемых данных, функции, предоставляемые сервисом, а также другую информацию, необходимую для взаимодействия клиента с этим сервисом. Формат WSDL-описания грид-сервиса включает: корневой элемент <definition>, в котором определяются пространства имен, используемых в данном описании; элемент <types>, включающий типы данных, используемые грид-сервисом; элемент <message>, описывающий сообщения, которые обрабатывает грид-сервис; элемент <portType>, содержащий описание операций, выполняемых грид-сервисом; список элементов <operation>, каждый из которых именует один из методов, выполняемых грид-сервисом, и описывает его входные и выходные сообщения; элементы <input>, <output> и <fault>, описывающие входные и выходные сообщения, а также сообщения об ошибках; элемент <binding>, определяющий для каждого сообщения конкретизацию сетевого протокола, способы кодировки и упаковки послания; элемент <service>, определяющий местонахождение сервиса и включающий один или несколько вложенных элементов <port>, каждый из которых задает адрес операции грид-сервиса.

Для запуска приложения в распределенной вычислительной среде нужно сформировать задание системе управления заданиями (СУЗ), используемой в этой среде. К таким системам относятся: во-первых, глобальные планировщики (метапланировщики) заданий типа GridWay [11]; во-вторых, локальные системы управления вычислительными кластерами, такие, например, как PBS [12] или Condor [13]. На сегодняшний день стандартом де-факто промежуточного программного обеспечения для организации грид-вычислений является пакет Globus Toolkit [14]. Поэтому будем называть СУЗ стандартизированной, если она входит в этот пакет или обеспечивает возможность взаимодействия с ним. Само задание представляет собой спецификацию процесса решения задачи, содержащую информацию о требуемых вычислительных ресурсах, исполняемых прикладных программах, входных/выходных данных, а также другие необходимые сведения.

Таким образом, при реализации сервис-ориентированного подхода к организации распределенных вычислений требуется решить две основные задачи: получить описание сервиса для приложения на языке WSDL и конвертировать пользовательский запрос к сервису в вычислительное задание для распределенной вычислительной среды.

2. Создание сервисов в инструментальном комплексе DISCENT

В ИДСТУ СО РАН разработан ряд высокоуровневых инструментальных средств [15], позволяющих в проблемно-ориентированных терминах как конструировать крупноблочные параллельные и распределенные вычислительные процессы, так и автоматически их синтезировать на основе непроцедурных постановок задач на вычислительной модели предметной области. Эти средства образуют фундаментальную основу для организации интеллектного сервис-ориентированного инструментария. В частности, инструментальный комплекс DISCENT реализует интеллектное управление вычислениями [16] в гетерогенной распределенной вычислительной среде, базируется на применении многоуровневых моделей знаний об этой среде и предоставляет унифицированный интерфейс взаимодействия прикладных систем модульного программирования на основе комбинирования различных способов их размещения (локальных и удаленных) и выполнения (параллельных, многовариантных, взаимосвязанных и др.). Определяющим фактором в дальнейшем его развитии является разработка средств оформления приложений в виде удаленно доступных сервисов.

Будем считать, что приложения, размещенные в узлах гетерогенной распределенной вычислительной среды, представлены в виде exe-файлов, реализованы для работы в пакетном режиме, а их конфигурационные параметры настроены для запуска этих приложений в соответствующих узлах. Зачастую в таком виде бывает представлено унаследованное программное обеспечение – программные системы или комплексы, не в полной мере соответст-

вующие характеристикам имеющейся распределенной вычислительной среды, но до сих пор используемые ввиду значительных финансовых, организационных, технических и прочих затруднений, связанных с их модернизацией или заменой.

Для каждого приложения генерируется программная оболочка, реализующая шаблоны вызовов типовых команд СУЗ, применяемых в вычислительной среде. Как правило, шаблон команды включает имя этой команды, а также списки ее опций и параметров, необходимых для работы с приложением. Процесс создания шаблонов рассматривается ниже.

3. Схема конвертирования пользовательских запросов к сервис-ориентированной среде в вычислительные задания

Схема конвертирования пользовательских запросов в вычислительные задания представлена на Рис. 1.

Администратор сервис-ориентированной среды с помощью веб-интерфейса модуля оформления приложений в виде сервисов добавляет описания приложений, размещенных в узлах вычислительной среды, в единый WSDL-файл. Этот файл описывает все операции, предоставляемые сервис-ориентированной средой. Для каждого приложения в форме добавления приложения заполняются следующие данные: имя и описания операции, входные/выходные параметры (имена, описание, типы данных), команда для запуска приложения при выполнении конкретной операции. Для каждого приложения может быть определено несколько операций. Таким образом, составляется шаблон, с помощью которого запрос пользователя будет преобразован в команду вызова удаленного приложения в конкретном узле вычислительной среды. Шаблоны сохраняются в базе данных. При обращении к грид-сервису запрос на выполнение операции конвертируются в SOAP-сообщение и поступает на SOAP-сервер, который вызывает программную оболочку. В процессе своего выполнения программная оболочка, в свою очередь, извлекает нужный шаблон вызова команды СУЗ из базы данных, конвертирует поступивший запрос и данные в задание для СУЗ. Далее планировщик СУЗ отправляет это задание на выполнение в распределенную вычислительную среду.

Рассмотрим пример программной оболочки для работы с приложением peispack.exe, реализующим поливариантную схему вычисления

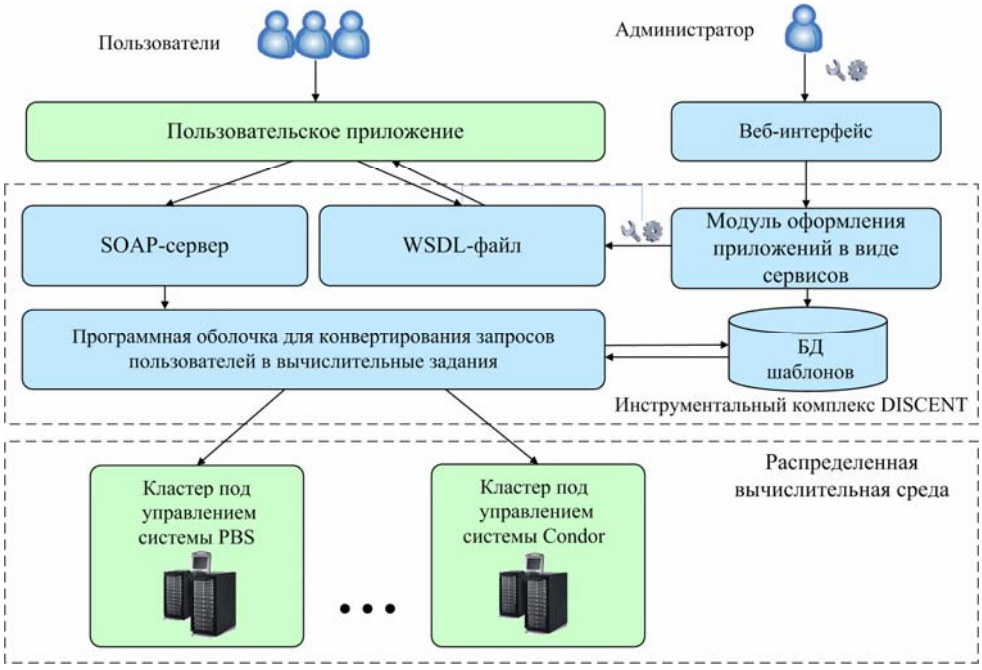

Рис. 1. Схема конвертирования пользовательских запросов к сервис-ориентированной среде в вычислительные задания

собственных значений произвольной плотной матрицы с использованием алгоритмов, представленных в работе [17]. Основные операции сервиса для этого приложения приведены на Рис. 2. Приложение получает на вход файл с входными данными, результаты счета так же выводятся в файл. Операция *RunPeispackExe* предназначена для запуска приложения peispack.exe. Данная операция в формате WSDL представлена двумя секциями (Рис. 3): секция *RunPeispackExeRequest* описывает параметры *file_input*, определяющий файл с входными данными, и *meta_scheduler*, задающий имя планировщика заданий (в текущей реализации в качестве планировщика заданий могут использоваться соответствующие утилиты систем

```
<portType name='MyServicePortType'>
    <operation name='RunPeispackExe'>
        <documentation>Usage peispack.exe [input.file].</documentation>
        <input message='tns:RunPeispackExeRequest'/>
        <output message='tns:RunPeispackExeResponse'/>
    </operation>
    <operation name='GetTasksList'>
        <documentation>Get task list</documentation>
        <input message='tns:GetTasksListRequest'/>
        <output message='tns:GetTasksListResponse'/>
    </operation>
    <operation name='GetTaskStatus'>
        <documentation>Get task status. Use Task ID</documentation>
        <input message='tns:GetTaskStatusRequest'/>
        <output message='tns:GetTaskStatusResponse'/>
    </operation>
    <operation name='GetTaskResult'>
        <documentation>Get task result. Use Task ID</documentation>
        <input message='tns:GetTaskResultRequest'/>
        <output message='tns:GetTaskResultResponse'/>
    </operation>
</portType>
```

Рис. 2. Фрагмент WSDL-файла, описывающий операции грид-сервиса

```
<message name='RunPeispackExeRequest'>
    <part name='file_input'                    type='xsd:file'/>
    <part name='meta_scheduler'                type='xsd:string'/>
</message>
<message name='RunPeispackExeResponse'>
    <part name='output' type='xsd:file'/>
</message>

<message name='GetTasksListResponse'>
    <part name='output_list' type='tns:stringArray'/>
</message>

<message name='GetTaskStatusRequest'>
    <part name='task_id_get_task_status' type='xsd:string'/>
</message>
<message name='GetTaskStatusResponse'>
    <part name='output_get_task_status' type='xsd:string'/>
</message>

<message name='GetTaskResultRequest'>
    <part name='task_id_get_task_result' type='xsd:string'/>
</message>
<message name='GetTaskResultResponse'>
    <part name='output_get_task_result' type='xsd:stringArray'/>
</message>
```

Рис. 3. Фрагмент WSDL-файла, описывающий параметры операций грид-сервиса

Condor и PBS, а также планировщик заданий инструментального комплекса DISCENT); секция *RunPeispackExeResponse* описывает параметр *output*, определяющий файл с результатами счета. Операция *RunPeispackExe* извлекает шаблон для запуска приложения peispack.exe (Рис. 4) из базы данных и конвертирует указанные пользователем данные в задание, которое передается планировщику заданий системы Condor. Данный планировщик выполняет планирование загрузки узлов распределенной вычислительной среды и осуществляет размещение и запуск полученного приложение в одном из них. Программная оболочка включает ряд дополнительных операций для работы с приложением: получения списка запущенных пользователем заданий (*GetTasksList*), их состояний выполнения (*GetTaskStatus*) и результатов счета (*GetTaskResult*).

Представленный на Рис. 4 шаблон для запуска приложения использует следующие переменные: $*ServiceName* – имя сервиса;

```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions name='{$ServiceName}'
 targetNamespace='http://example.org/{$ServiceName}'
 xmlns:tns=' http://example.org/{$ServiceName} '
 xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
 xmlns:xsd='http://www.w3.org/2001/XMLSchema'
 xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
 xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
 xmlns='http://schemas.xmlsoap.org/wsdl/'>

<message name='{$ServiceOperationName}Request'>
    <part name='{$InParam1}' type='{$InParamType1}'/>
    <part name='{$InParam2}' type='{$InParamType2}'/>
    <part name='{$InParamN}' type='{$InParamTypeN}'/>
</message>
<message name='{$ServiceOperationName}Response'>
    <part name='{$OutParam1}' type='{$OutParamType1}'/>
    <part name='{$OutParam2}' type='{$OutParamType2}'/>
    <part name='{$OutParamN}' type='{$OutParamTypeN}'/>
</message>

<portType name='{$ServiceName}PortType'>
    <operation name='{$ServiceOperationName}'>
        <documentation>{$Service_description}</documentation>
        <input message='tns:{$ServiceOperationName}Request'/>
        <output message='tns:{$ServiceOperationName}Response'/>
    </operation>
</portType>
<binding name='{$ServiceName}Binding' type='tns:{$ServiceName}PortType'>
    <soap:binding style='rpc'
    transport='http://schemas.xmlsoap.org/soap/http'/>
    <operation name='{$ServiceOperationName}'>
        <soap:operation soapAction='urn:xmethods-delayed-quotes#{$ServiceOperationName}'/>
        <input>
            <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
            encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
        </input>
        <output>
            <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
            encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
        </output>
    </operation>
</binding>

<service name='{$ServiceName}'>
    <port name='{$ServiceName}Port' binding='{$ServiceName}Binding'>
        <soap:address location='http://{$Domain}/{$ServiceName}/{$ServiceName}.php'/>
    </port>
</service>
</definitions>
```

Рис. 4. Шаблон для запуска приложения

$*ServiceOperationName* – имя операции, $*InParam1*, …, $*InParamN* – входные параметры, $*InParamType1*, …, $*InParamTypeN* – типы входных параметров, $*OutInParam1*, …, $*OutParamN* – выходные параметры, $*OutInParamType1*, …, $*OutParamTypeN* – типы выходных параметров, $*Service_description* – описание операции, $*Domain* – домен, в котором размещен сервис.

Описание грид-сервисов может быть использовано в дальнейшем в инструментариях, обеспечивающих организацию веб-ориентированного доступа к грид-сервисам и поддерживающих формат WSDL. При необходимости использования услуг грид-сервисов в удаленных пользовательских приложениях пользователь может скачать WSDL-файл с описанием грид-сервисов и реализовать функции анализа этого файла и вызова нужных операций грид-сервисов в своем приложении.

4. Конкретизация вычислительных заданий

Средства конкретизации вычислительных заданий [18] разработаны для более детальной настройки требований к вычислительной среде, содержащихся в заданиях, с целью обеспечения более эффективного планирования и распределения ресурсов планировщиками СУЗ при обработке этих заданий. Схема взаимодействия данных средств между собой и с СУЗ представлена на Рис. 5.

Система виртуальной декомпозиции ресурсов позволяет администратору распределенной вычислительной среды классифицировать задания, которые могут поступать в эту среду, по их характеристикам и определить для каждого ресурса этой среды те классы заданий, которые по своим характеристикам наиболее подходят вычислительным возможностям этих ресурсов.

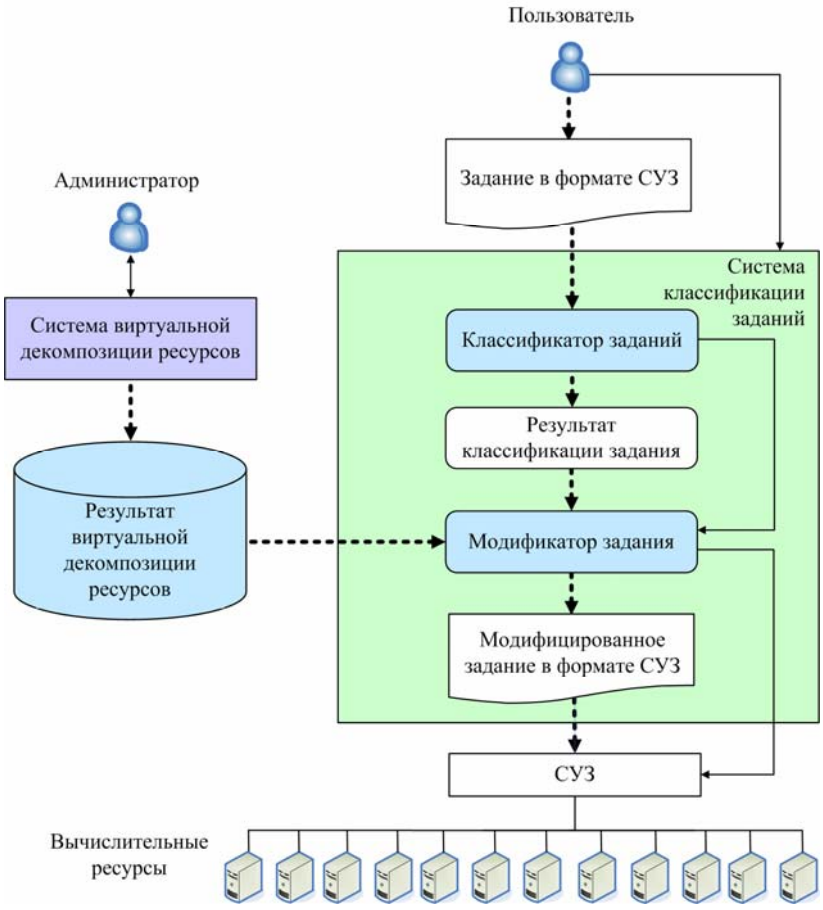

Рис. 5. Схема взаимодействия систем виртуальной декомпозиции ресурсов и классификации заданий

Результат виртуальной декомпозиции ресурсов сохраняется в базе данных.

Система классификации заданий осуществляет перехват заданий, поступающих в распределенную вычислительную среду. Данная система включает две основные подсистемы: классификатор задания и модификатор задания. Классификатор задания выполняет унификацию задания и определение дополнительных его характеристик, извлекаемых из переменных окружения вычислительной среды, конфигурационных параметров СУЗ или статистических файлов (системных и/или пользовательских). Затем он определяет класс задания и передает исходное задание и результат его классификации модификатору задания. Модификатор задания добавляет в задание дополнительные параметры, определяющие множество ресурсов, соответствующих классу задания, и передает модифицированное задание в СУЗ. В дальнейшем планировщик СУЗ будет выбирать ресурсы для выполнения задания только из множества ресурсов, заданного дополнительными параметрами модифицированного задания.

Проведенные эксперименты [18, 19] по применению средств конкретизации вычислительных заданий в процессе управления разнородной распределенной вычислительной средой показывают возможность существенного улучшения целого ряда важных показателей ее функционирования, таких как характеристики обслуживания очередей заданий, коэффициенты надежности вычислений, эффективности выделения и полезного использования вычислительных ресурсов.

Заключение

В целом в статье представлен подход к организации сервис-ориентированных распределенных вычислений с помощью инструментального комплекса DISCENT. Реализация рассмотренного в статье подхода позволит представить прикладное программное обеспечение, разрабатываемое в ИДСТУ СО РАН, в виде сервисов, доступных широкому кругу конечных пользователей. В частности, в настоящее время ведутся работы по представлению в виде сервисов ряда функциональных возможностей инструментального комплекса РЕБУС [20, 21], ориентированного на применение в фундаментальных и прикладных исследованиях при проведении вычислительных экспериментов в разнообразных предметных областях, где естественным образом возникают дискретные модели в виде систем булевых уравнений.

Следует отметить следующие отличительные особенности инструментального комплекса DISCENT для оформления приложений в виде сервисов: возможность использования в качестве приложений унаследованного программного обеспечения без каких-либо его изменений; возможность запуска приложений в произвольных узлах любой распределенной вычислительной среды, использующей для управления заданиями стандартизированные СУЗ; наличие системы планирования вычислений [6, 16], обеспечивающей построение схем выполнения взаимосвязанных заданий, что делает возможным реализацию комбинированного применения сервисов; наличие средств виртуальной декомпозиции ресурсов и классификации заданий, позволяющих выполнять вычислительные задания с помощью ресурсов, наиболее подходящих для этого с точки зрения администратора распределенной вычислительной среды.

Литература

1. Foster I. Service-Oriented Science // Science. 2005. Vol. 308. No. 5723. Pp. 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 Всерос. суперкомпьютерной конф. М.: Изд-во МГУ, 2009. С. 463-467.
3. Бухановский А.В., Ковальчук С.В., Марьин С.В. Интеллектуальные высокопроизводительные программные комплексы моделирования сложных систем: концепция, архитектура и примеры реализации // Известия вузов. Приборостроение. 2009. Т. 52. № 10. С. 5-24.
4. Жижимов О.Л., Пестунов И.А., Федотов А.М. Структура сервисов управления метаданными для разнородных информационных систем // Электронные библиотеки: российский научный электронный журнал. 2012. Т. 15. № 5.
5. Шокин Ю.И., Федорук М.П., Чубаров Д.Л., Юрченко А.В. О развитии инфраструктуры суперкомпьютерных и распределенных вычислений в СО РАН // Информа-

ционные технологии и вычислительные системы. 2011. № 3. С. 9-19.
6. Бычков И.В., Корсуков А.С., Опарин Г.А., Феоктистов А.Г. Инструментальный комплекс для организации гетерогенных распределенных вычислительных сред // Информационные технологии и вычислительные системы. 2010. № 1. С. 45-54.
7. Iverson W. Real world Web services. O'Reilly, 2004. 207 p.
8. Kundu P., Das D., Ratha B. WSDL Specification of Services for Service Oriented Architecture (SOA) Based Implementation of a CRM Process // International Journal of Scientific and Engineering Research. 2012. Vol. 3. No. 10. Pp. 1-24.
9. Walsh A. UDDI, SOAP, and WSDL: The Web Services Specification Reference Book. Pearson Education, 2002. 305 p.
10. Черняк Л. Web-сервисы, grid-сервисы и другие // Открытые системы. 2004. № 1. С. 20-27.
11. Herrera J., Huedo E., Montero R., Llorente I. Porting of Scientific Applications to Grid Computing on GridWay // Scientific Programming. 2005. Vol. 13. No. 4. Pp. 317–331.
12. Henderson R. Job scheduling under the portable batch system // Job scheduling strategies for parallel processing. Springer, 1995. Pp. 279-294.
13. Litzkow M., Livny M., Mutka M. Condor – A Hunter of Idle Workstations // In 8th International Conference of Distributed Computing Systems (ICDCS). IEEE CS Press, Los Alamitos, CA, USA, 1988. Pp. 104-111.
14. Foster I. Globus Toolkit Version 4: Software for Service-Oriented Systems // IFIP International Conference on Network and Parallel Computing. Springer, 2006. Pp. 2-13.
15. Бычков И.В., Опарин Г.А., Новопашин А.П., Феоктистов А.Г., Корсуков А.С., Сидоров И.А. Высокопроизводительные вычислительные ресурсы ИДСТУ СО РАН: Текущее состояние, возможности и перспективы развития // Вычислительные технологии. 2010. Т. 15. № 3. С. 69-81.
16. Бычков И.В., Опарин Г.А., Феоктистов А.Г., Корсуков А.С. Децентрализованное управление потоками заданий в интегрированной кластерной системе // Вестник НГУ. Серия: Информационные технологии. – 2011. Т. 9. Вып. 2. С. 42-54.
17. Уилкинсон Дж.Х., Райнш К. Справочник алгоритмов на языке АЛГОЛ. М.: Машиностроение, 1976. 389 с.
18. Бычков И.В., Опарин Г.А., Феоктистов А.Г., Корсуков А.С. Распределение заданий в интегрированной кластерной системе на основе их классификации // Вычислительные технологии. 2013. Т. 18. № 2. С. 25–32.
19. Бычков И.В., Опарин Г.А., Феоктистов А.Г., Корсуков А.С. Испытание и оценка надежности интегрированных кластерных систем на основе их комплексного моделирования // Вестник компьютерных и информационных технологий. 2013. № 3. С. 3-8.
20. Опарин Г.А., Богданова В.Г. РЕБУС – интеллектуальный решатель комбинаторных задач в булевых ограничениях // Вестник НГУ. Серия: Информационные технологии. 2008. Т. 6. Вып. 1. С. 60-68.
21. Опарин Г.А., Богданова В.Г. Инструментальные средства автоматизации параллельного решения булевых уравнений на многоядерных процессорах // Программные продукты и системы. 2012. № 1. С. 10-14.

Бычков Игорь Вячеславович. Директор Института динамики систем и теории управления СО РАН. Окончил Иркутский государственный университет в 1983 году. Академик РАН, доктор технических наук, профессор. Автор 141 печатной работы и 8 монографий. Область научных интересов: ГИС, Grid и веб-технологии. E-mail: idstu@icc.ru

Опарин Геннадий Анатольевич. Заместитель директора по научной работе Института динамики систем и теории управления СО РАН. Окончил Казанский авиационный институт им. А.Н.Туполева в 1974 году. Доктор технических наук, профессор. Автор 203 печатных работ. Область научных интересов: модели и методы организации распределенных вычислений, Grid. E-mail: idstu@icc.ru

Феоктистов Александр Геннадьевич. Старший научный сотрудник Института динамики систем и теории управления СО РАН. Окончил Иркутский государственный университет в 1987 году. Кандидат технических наук, доцент. Автор 89 печатных работ. Область научных интересов: методы и инструментальные средства организации распределенных вычислений, Grid. E-mail: agf@icc.ru

Корсуков Александр Сергеевич. Младший научный сотрудник Института динамики систем и теории управления СО РАН. Окончил Иркутскую государственную экономическую академию в 2004 году. Кандидат технических наук. Автор 24 печатных работ. Область научных интересов: инструментальные средства организации распределенных вычислений, Grid, веб-технологии. E-mail: alexask@mail.ru

Богданова Вера Геннадьевна. Старший научный сотрудник Института динамики систем и теории управления СО РАН. Окончила Иркутский государственный университет в 1979 году. Кандидат технических наук, доцент. Автор 59 печатных работ. Область научных интересов: методы и инструментальные средства организации распределенных вычислений, Grid. E-mail: bvg@icc.ru»