УДК 004.75 DOI: 10.21782/GIPR0206-1619-2016-6(160-163)

Р. К. ФЁДОРОВ[1], А. С. ШУМИЛОВ[2]

[1] Иркутский научный центр СО РАН, 664033, Иркутск, ул. Лермонтова, 134, Россия, fedorov@icc.ru
[2] Институт динамики систем и теории управления им. В. М. Матросова СО РАН, 664033, Иркутск, ул. Лермонтова, 134, Россия, alexshumilov@yahoo.com

ЗАДАНИЕ ГРАФА ЗАВИСИМОСТЕЙ ДЛЯ КОМПОЗИЦИЙ СЕРВИСОВ С ПОМОЩЬЮ JAVASCRIPT СЦЕНАРИЕВ

В современном мире информационных технологий бурно развивается область сервис-ориентированных вычислений. Часто возникают задачи, для решения которых необходимо использовать несколько сервисов. Объединение нескольких сервисов для решения какой-либо задачи называется композицией сервисов. Зависимости между сервисами обычно описываются направленным ациклическим графом. Существует большое количество средств задания таких композиций, в основном это графические программные средства и различные стандарты разметки. Однако часто возникает ситуация, когда нужно задать композицию сервисов как обычную программу, имея при этом возможность обработки промежуточных результатов работы сервисов с последующим извлечением графа зависимостей сервисов по данным.

В рамках Геопортала Института динамики систем и теории управления (ИДСТУ) СО РАН для выполнения композиций распределенных сервисов был разработан способ задания композиций сервисов с помощью программ на языке JavaScript. Для задания композиции сервисов пользователь внутри сценария вызывает доступные распределенные сервисы с помощью специальных функций-оберток, как обычных функций языка. Внутри сценария пользователь может обрабатывать результаты работы сервисов, а также строить сложные управляющие конструкции с помощью стандартных средств языка. При обработке и выполнении таких сценариев происходит автоматическое построение графа зависимостей сервисов по данным, что впоследствии может быть использовано, например, для построения оптимального расписания выполнения сервисов в распределенной среде.

Способ задания сценариев сервисов с помощью программ на языке JavaScript был успешно апробирован на реальных задачах и интегрирован в Портал ИДСТУ СО РАН.

Ключевые слова: *сервис-ориентированная архитектура, композиция сервисов, JavaScript, SOA, DAG.*

R. K. FEDOROV[1] AND A. S. SHUMILOV[2]

[1] Irkutsk Scientific Center SB RAS, 664033, Irkutsk, Lermontova str., 134, Russia, fedorov@icc.ru
[2] V. M. Matrosov Institute for System Dynamics and Control Theory SB RAS, 664033, Irkutsk, Lermontova str., 134, Russia, alexshumilov@yahoo.com

DEFINING THE TASK OF DEPENDENCY GRAPH FOR SERVICE COMPOSITIONS USING THE JAVASCRIPT SCENARIOS

The field of service-oriented computations is actively developed in today's world of information science. Tasks, which require use of multiple services, arise constantly. The combination of multiple services for the solution of a problem is called service composition. Dependencies between services can be usually described with the directed acyclic graph (DAG). There are a lot of ways to define service compositions; most of them use the graphic interface software or various markup standards. However, it is a common situation when it is convenient to define the composition of services as a piece of program code, at the same time the processing of intermediate service results and extraction of service dependencies graph should be available.

The method of defining service compositions as the JavaScript programs was designed in ISDCT SB RAS in order to execute service compositions. In order to define the composition, user calls services inside of the JavaScript code as regular functions. The intermediate processing of service results and building of complex control structures with standard programming language are available. During the processing and execution of scenarios the services dependencies graph is automatically built. The built graph can be later used, for example, for finding of optimal schedule for services execution in the distributed environment.

The method of defining the services composition scenarios with JavaScript programs was successfully tested with real tasks and was integrated into the Portal of ISDCT SB RAS.

Keywords: *service-oriented architecture, service composition, JavaScript, SOA, DAG.*

ВВЕДЕНИЕ

В последнее время активно развивается сервис-ориентированная архитектура вычислений (Service-Oriented Architecture, SOA) [1], которая предполагает представление программных компонентов в виде

отдельных вычислительных сервисов. Основными преимуществами данного подхода является простота использования сервисов (сложность реализации сервиса полностью скрывается за его интерфейсом), сравнительно легкое тестирование и отладка (так как тестирование производится для каждого сервиса отдельно), масштабируемость и возможность повторного использования.

В области обработки пространственных данных все больше вычислений производится с помощью распределенных сервисов, т. е. сервисов, развернутых на разных вычислительных узлах. Например, активно используются сервисы геокодирования, которые позволяют по адресу получать координаты, и сервисы построения маршрутов, дающие возможность определить кратчайший путь с учетом дорожной сети, их покрытия и т. д.

Основные преимущества SOA, описанные ранее, оптимально проявляются при использовании композиций сервисов, где решение сложной задачи обеспечивается взаимодействием сервисов и обменом данными между ними. Существует несколько подходов к заданию композиции сервисов, которые сводятся к определению графа зависимостей вызовов сервисов.

ОБЗОР СРЕДСТВ

Композицией называется набор сервисов с определенными между ними зависимостями, решающий какую-либо задачу или автоматизирующий какой-либо процесс. Выполнение сервиса в контексте теории расписаний называется заданием (требованием). Общепризнанным стандартом является определение зависимостей между заданиями с помощью направленного ациклического графа (Directed Acyclic Graph, DAG) [2, 3], в котором вершинами являются сами задания, а ребра показывают зависимости между заданиями по данным.

Рассматривая подходы к заданию композиций сервисов, необходимо выделить два основных вида таких средств — графические и текстовые.

Одним из наиболее известных текстовых способов описания композиций сервисов являются стандарты BPEL (Business Process Execution Language) [4] и XPDL (XML Process Definition Language) [5]. Оба способа дескриптивно задают граф зависимостей заданий DAG. Несмотря на то что стандарт XPDL был принят раньше, чаще всего в работе используется стандарт BPEL, так как BPEL более приспособлен для описания взаимодействия сервисов. В свою очередь, XPDL позволяет использовать сервисы, не имеющие веб-интерфейса и определять графическое представление заданного взаимодействия. Данные стандарты широко используются, но сложны в написании, для них требуется знание как языка разметки XML, так и конструкций самого стандарта.

Одними из наиболее популярных графических средств задания композиций сервисов являются программные пакеты UNICORE (UNiform Interfaceto COmputing REsources) и Taverna.

UNICORE — программный пакет, предназначенный для организации вычислений в сервис-ориентированных средах [6]. Архитектура UNICORE представлена тремя уровнями: пользователя, сервисов и системный уровень. UNICORE имеет удобный интерфейс, возможность масштабирования инструментального комплекса; возможность добавления встроенных обработчиков на поддерживаемых языках программирования; наличие конструкций while, foreach и if. Однако эти управляющие конструкции позволяют делать условные переходы только на основе значений внутренних переменных или значений выход-статусов отработавших программ. Нельзя сделать условный переход на основе результирующих данных работы сервисов.

Для задания композиций сервисов также широко известен программный продукт Taverna [7], позволяющий с помощью графического интерфейса определять последовательности выполнения заданий. Taverna использует функциональную модель управления данными. Taverna распространен в научной среде и часто применяется в астрономии, биоинформатике (например, для определения генов, ответственных за конкретное заболевание [8]). Taverna поддерживает парадигму событийно-ориентированного программирования, что предоставляет широкие возможности описания иерархических схем алгоритмов. Но для разработки схем, включающих более сложные структуры (например, циклы), данного аппарата оказывается недостаточно.

Таким образом, можно отметить, что существуют подходы к заданию композиции сервисов, где граф зависимостей заданий DAG полностью определяется до момента выполнения, причем пользователю необходимо либо использовать графический интерфейс для нахождения зависимостей между сервисами, либо вручную описывать зависимости между сервисами с помощью языков разметки. Однако не существует такого средства, которое бы позволяло описать композицию сервисов в виде программы на одном из языков программирования, из которой впоследствии автоматически извлекался бы граф зависимости сервисов по данным для последующей обработки.

ОПРЕДЕЛЕНИЕ КОМПОЗИЦИИ СЕРВИСОВ С ПОМОЩЬЮ JAVASCRIPT

В ИДСТУ СО РАН, в рамках Интеграционной программы ИНЦ СО РАН, был разработан и реализован способ задания композиций сервисов с помощью языка программирования JavaScript. Композиция сервисов представляет собой программный код, в котором вызов сервисов осуществляется с помощью специальных функций, закрепленных за каждым сервисом. Применение процедурного языка для определения композиции сервисов позволяет задавать сложные алгоритмы с использованием ветвлений, циклов, рекурсий и других конструкций языка программирования JavaScript.

Участвующие в сценариях сервисы должны быть предварительно зарегистрированы в каталоге Портала и могут находиться на любом вычислительном узле независимо от его местоположения и характеристик. Таким образом, Порталу известно только их сетевое расположение, описание параметров и характеристики самих вычислительных узлов.

Граф зависимостей заданий формируется динамически в процессе выполнения JavaScript-кода и зависит от начальных данных или промежуточных результатов. Для вызова сервисов используются так называемые функции-обертки, каждая из которых соответствует определенному сервису. Во время вызова функций-оберток создаются вершины графа, т. е. регистрируются задания. Для определения зависимостей между заданиями используются объекты класса *ValueStore*, которые передаются в качестве параметров функций сервисов. Объекты класса *ValueStore* необходимы для однозначной идентификации передаваемых данных при вызове функций сервисов. Вызовы специальных функций не блокируют выполнение JavaScript-кода. Блокировка сценария производится только в случае вызова метода *Get* объекта *ValueStore*, если данные еще не вычислены, с помощью него можно получить значения данных.

Применение процедурного языка программирования для определения DAG имеет ряд преимуществ:

– для большинства программистов это привычный язык разработки алгоритмов, так как применение сервисов не отличается от использования различных программных библиотек;

– наличие готовых алгоритмов, которые можно применить в этом подходе для промежуточной обработки результатов работы сервисов;

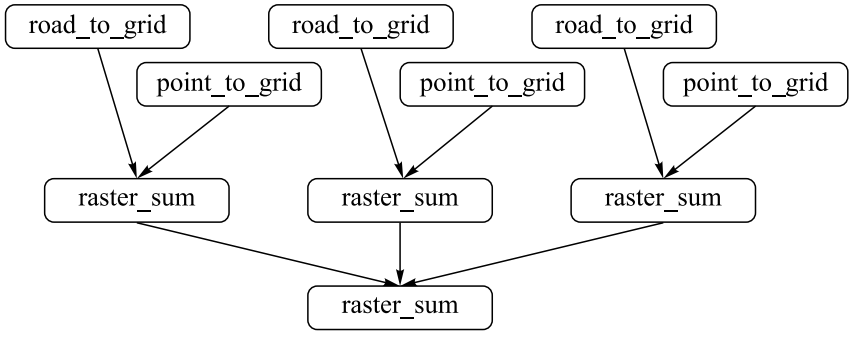
– определение зависимостей заданий происходит динамически.

АПРОБАЦИЯ

В целях апробации предложенного подхода к планированию выполнения композиции с зависимостями между сервисами по данным была выбрана задача расчета загрязнения тремя парниковыми газами для определенного региона.

В сценарии используется три различных сервиса — *vectorToGrid*, *roadToGrid*, *raster_sum*. Сервис *vectorToGrid* применяется для расчета выделяемых загрязнений от точечных источников (котельных, ТЭЦ и др.) в ячейках регулярной сетки; сервис *roadToGrid* — для расчета выделяемых загрязнений от автотранспорта в ячейках регулярной сетки; сервис *raster_sum* — для суммирования выделяемых загрязнений от точечных источников и автотранспорта.

На рисунке приведен составленный в результате анализа сценария граф зависимости сервисов по данным (DAG).



Граф зависимости сервисов по данным (DAG).

ЗАКЛЮЧЕНИЕ

Спроектирован и разработан, а также интегрирован в существующую инфраструктуру Портала ИДСТУ СО РАН способ задания композиций сервисов в виде сценариев на языке JavaScript. Основным преимуществом разработанного способа является возможность задания композиций в виде стандартных программ на распространенном языке программирования JavaScript. Пользователь при написании сценария имеет дело с результатами работы сервисов как с обычными переменными, в то же время пользуясь всеми возможностями языка — как для промежуточной обработки полученных данных, так и для построения сложных управляющих конструкций внутри сценария.

В результате апробаций отмечено, что способ задания композиций сервисов в виде сценария на языке программирования JavaScript успешно решает проблему составления графа зависимости сервисов по данным при минимальных временных и трудовых затратах пользователя на построение непосредственно сценариев сервисов.

Работа выполнена в рамках Интеграционной программы ИНЦ СО РАН «Фундаментальные исследования и прорывные технологии как основа опережающего развития Байкальского региона и его межрегиональных связей».

СПИСОК ЛИТЕРАТУРЫ

1. **Emig C., Weisser J., Abeck S.** Development of SOA-Based Software Systems — an Evolutionary Programming Approach // Advanced Intern. Conference on Telecommunications and Intern. Conference on Internet and Web Applications and Services (AICT-ICIW'06). — Guadeloupe, French Caribbean, 2006. — P. 14–15.
2. **Sachdeva S., Rana P.** A review of multiprocessor Directed Acyclic Graph (DAG) scheduling algorithms // Intern. Journ. Computer Science & Communication. — 2015. — Vol. 66, N 11. — P. 67–72.
3. **Sinnen O.** Task scheduling for parallel systems. — Hoboken, New Jersey, United States: Wiley-Interscience, 2007. — 108 p.
4. **OASIS** Web Services Business Process Execution Language (WSBPEL) TC // OASIS Consortium [Электронный ресурс]. — https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel (дата обращения 25.06.2016).
5. **XML** Process Definition Language (XPDL) // XPDL definitions [Электронный ресурс]. — http://www.xpdl.org/ (дата обращения 24.09.2016).
6. **UNICORE** distributed computing // UNICORE official website [Электронный ресурс]. — https://www.unicore.eu/ (дата обращения 26.09.2016).
7. **Taverna** Workflow System // Taverna System [Электронный ресурс]. — https://taverna.incubator.apache.org/ (дата обращения 26.06.2016).
8. **Oinn T., Addis M., Ferris J.** Taverna: a tool for the composition and enactment of bioinformatics workflows // Bioinformatics. — 2004. — Vol. 20. — P. 82–98.

Поступила в редакцию 21 октября 2016 г.