Институт вычислительных технологий СО РАН
Институт автоматики и электрометрии СО РАН
Федеральный научно-производственный центр «Алтай»
Алтайский государственный университет
Новосибирский государственный технический университет

Всероссийская конференция

«Обработка пространственных данных в задачах мониторинга природных и антропогенных процессов (SDM – 2015)»

24 – 28 августа 2015 года

Сборник трудов

Новосибирск, 2015

УДК 004

Издание осуществляется при финансовой поддержке
Российского фонда фундаментальных исследований
(грант № 15-07-20719-г)

Обработка пространственных данных в задачах мониторинга природных и антропогенных процессов [Электронный ресурс]: Сборник трудов всероссийской конференции (24-28 августа 2015 г., с. Усть-Сема, Республика Алтай). Новосибирск: ИВТ СО РАН, 2015. 259 с.: 100 экз. ISBN 978-5-905569-10-4.

Сборник трудов подготовлен по результатам работы всероссийской конференции «Обработка пространственных данных в задачах мониторинга природных и антропогенных процессов» (SDM-2015), которая проходила с 24 по 28 августа 2015 г. на базе ФНПЦ «Алтай» (с. Усть-Сема, Республика Алтай). В сборнике представлены результаты исследований по следующим направлениям: интегрированные геоинформационные технологии и системы для задач мониторинга, оперативный региональный спутниковый мониторинг окружающей среды, моделирование экологических и техногенных процессов и систем. Сборник будет полезен для научных и инженерных работников, аспирантов и студентов вузов, занимающихся проблемами мониторинга окружающей среды.

ПРОГРАММНЫЙ КОМИТЕТ КОНФЕРЕНЦИИ

ISBN 978-5-905569-10-4

АСИНХРОННОЕ ВЫПОЛНЕНИЕ WPS СЕРВИСОВ НА ГЕТЕРОГЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСАХ ОБЛАЧНОЙ ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНОЙ СРЕДЫ

Бычков И.В., Ружников Г.М., Потапов В.П., Шумилов А.С., Фёдоров Р.К.
Институт динамики систем и теории управления им. В.М. Матросова, Иркутск

С развитием геоинформационных технологий объемы информации, обрабатываемой различными вычислительными сервисами, постоянно увеличиваются. Часто возникают задачи, для решения которых необходимо совместно использовать несколько сервисов. В ИДСТУ СО РАН был разработан способ интеграции гетерогенных вычислительных сервисов друг с другом на основе сценариев на языке программирования JavaScript. Данная работа исследует проблему ускорения работы сценариев путем моделирования и реализации асинхронного вызова сервисов с обеспечением контроля передаваемых между сервисами данных.

Введение. В последнее время в информационных технологиях наблюдается как постоянное увеличение объемов обрабатываемой информации, так и появление все большего числа вычислительных сервисов, обрабатывающих различные данные. Одной из быстрорастущих областей информационных технологий является обработка пространственных данных, в силу как роста качества пространственных данных, так и увеличения сложности задач и алгоритмов, используемых при работе с пространственными данными.

В силу увеличения сложности разрабатываемых геоинформационных сервисов часто возникает необходимость интеграции сервисов друг с другом, то есть обеспечения обмена данными между сервисами для решения какой-либо задачи. Однако, для решения комплексных задач, возникающих перед коллективами ученых, организации простых цепочек сервисов с последовательной передачей данных между сервисами часто недостаточно. В связи с этим возникает необходимость в программных средствах, позволяющих интегрировать сервисы друг с другом с использованием сложных логических конструкций, обработкой передаваемых данных, управлением потока выполнения и т.д. Одной из основных проблем, с которыми сталкиваются разработчики программных систем для интеграции веб-сервисов, является достаточно долгое время выполнения интегрированных сервисов, так как отдельно выполняющиеся сервисы могут выполняться до нескольких месяцев.

В ИДСТУ СО РАН был разработан способ интеграции вычислительных сервисов друг с другом на основе сценариев на языке программирования JavaScript, в котором вызов сервисов представляет собой вызов специальных функций на языке JavaScript [1]. Специальные функции реализуют протокол взаимодействия с сервисами для передачи параметров, запуска, получения результата и т.д. Функции являются блокирующими, т.е. они возвращают управление при окончании работы сервиса. Соответственно при выполнении сценария в один момент времени выполняет только один сервис. Однако при решении многих задач возможно одновременное выполнение нескольких сервисов, что позволит значительно ускорить выполнение сценариев. Таким образом, данная работа исследует способы ускорения выполнения сценариев.

Обзор существующих технологий. Одним из наиболее развитых и популярных решений для интеграции вычислительных сервисов является Oracle BPEL Process Manager – программный продукт корпорации Oracle, предлагающий построение цепочек выполнения вычислительных сервисов с помощью графического. Все операции, описывающиеся с помощью графических инструментов, записываются в виде XML документов, соответствующих

стандарту BPEL (Business Process Execution Language) [3] и впоследствии выполнимых с помощью специального BPEL процессора. Oracle BPEL Process Manager поддерживает как большое количество интерфейсов вычислительных сервисов и источников данных (WSDL, Oracle сервисы, базы данных, файловые системы, веб-доступные ресурсы), так и большинство операций, определенных в стандарте BPEL – логические конструкции, возможность асинхронного выполнения сервисов, обработка передаваемых между сервисами данных. Однако, данный программный пакет не предоставляет инструментов для автоматического построения интерфейса ввода и вывода данных для запускаемых сервисов, что заметно осложняет работу для неопытного пользователя, а также требует знания XML и BPEL.

Другим средством интеграции сервисов друг с другом является программная система Taverna. Данная система также предоставляет графический интерфейс для интеграции сервисов. Taverna использует свой собственный формат для описания интеграции сервисов друг с другом и делает больший акцент на обработке передаваемых между сервисами данных, чем на управлении непосредственно участвующими вычислительными сервисами. Говоря о возможностях построения интеграции сервисов, стоит отметить, что Taverna также поддерживает асинхронное выполнение сервисов. Несмотря на ориентированность на биотехнологии, рассматриваемый программный продукт успешно используется в других областях науки [2].

В результате проведенного анализа существующих решений наиболее развитым подходом является использование BPEL, который позволяет выполнять несколько сервисов одновременно. Использование языка JavaScript по сравнению с BPEL и другими подходами имеет ряд преимуществ: скорость выполнения сценария, мощность процедурного языка (объектно-ориентированное программирование, функциональные типы и т.д.), наличие готовых библиотек, реализующих различные алгоритмы и структуры данных, лаконичность и т.д.

На текущий момент считается, что логика работы сценариев будет достаточно простой. Поэтому не уделяется внимание уменьшению сложности разработки алгоритмов сценариев и скорости их выполнения. Реализация асинхронного выполнения сценариев на языке JavaScript является перспективным направлением исследования.

Сценарии вычислительных сервисов. В ИДСТУ СО РАН был разработан способ интеграции вычислительных сервисов друг с другом на основе языка программирования JavaScript. Предложенный способ отличается от рассмотренных выше способов тем, что он использует всю мощь логических и управляющих конструкций языка JavaScript, позволяет передавать между сервисами такие сложные данные, как файлы и данные из реляционных таблиц, а также поддерживает неограниченное время выполнения сервисов. Интеграция осуществляется с помощью сценариев, представляющих из себя программы на JavaScript, в которых зарегистрированные в системе распределенные сервисы вызываются с помощью специальных функций-оберток. Параметры в обертки передаются как в обычных вызовах JavaScript функций. Среди доступных способов обращения к вычислительным сервисам, таких как WSDL, был выбран WPS в силу его ориентированности на обработку пространственных данных.

Основываясь на результатах анализа существующих решений, а также принимая во внимание специфику используемого способа интеграции сервисов в виде JavaScript-сценариев, решение проблемы ускорения выполнения сценариев является разработка модели асинхронного вызова сервисов, а также реализация и апробация асинхронного вызова сервисов. В рамках этой модели необходимо упростить разработчику реализацию асинхронной модели выполнения сценария.

Среда выполнения сценариев вычислительных сервисов. Как уже было сказано ранее, сценарием вычислительных сервисов называется программа на языке JavaScript, которая имеет в своем теле вызовы каких-либо заранее известных распределённых сервисов, доступных по одному из протоколов (WSDL, WPS и т.д.). Сервисы внутри программы вызываются с помощью специальных функций-оберток, принимающих соответствующие параметры для каждого сервиса.

Механизм вызова зарегистрированных в системе вычислительных сервисов с помощью уникальных функций-оберток работает следующим образом – перед тем, как послать сценарий в модуль выполнения, подсистема запуска сценариев определяет участвующие в сценарии сервисы и производит генерацию кода. Далее сгенерированный код регистрирует функции-обертки в глобальном контексте выполнения, таким образом, их можно вызывать из любого места сценария.

При запуске какой-либо функции-обертки модуль выполнения на основании параметров сервиса автоматически выполняет запросы к удаленным серверам на выполнение сервисов, обрабатывает промежуточные результаты, возвращает результаты выполнения.

На рисунке 1 приведена схема очередности запросов при синхронном вызове сервисов для приведенного сценария

```
function pollution_test(input) {
   var result1 = service_1(input.a);
   var result2 = service_2(input.b);
   return service_3(result1, result2);
}
```

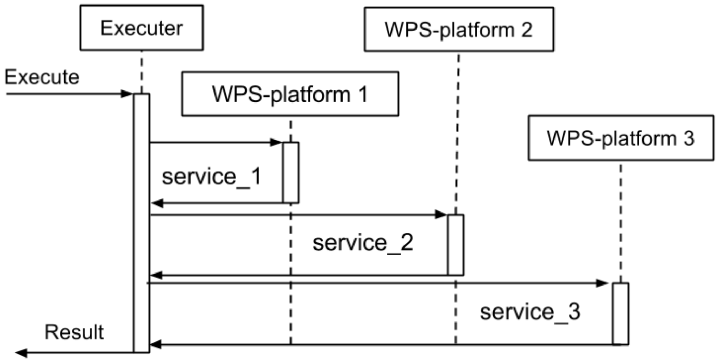

Рис. 1. Синхронный вызов сервисов.

Асинхронное выполнение сервисов. Основной идей асинхронной модели выполнения сценария является вызов сервисов сразу при готовности входных данных. При этом сценарий должен автоматически формировать очередь выполнения сервисов. Порядок выполнения сервисов должен формировать на основе зависимости по данным. Если сервис на входе использует результаты другого сервиса, то соответственно он должен ждать окончания его работы. Выполнение функций вызова сервисов не должно быть блокирующим, то есть как только JavaScript-обертка сервиса вызвана, управление сразу возвращается в сценарий.

Для реализации программного решения для поставленной проблемы сначала необходимо составить математическую модель асинхронного вызова сервисов в сценарии распределенных сервисов.

Обозначим

$S_{1..n}$ – множество участвующих в сценарии сервисов;

S_{ij} – экземпляр *j* участвующего в сценарии сервиса S_i;

IN_{Sij} – множество входных параметров экземпляра сервиса S_{ij};

OUT_{Sij} – множество результатов выполнения экземпляра сервиса S_{ij};

Ready(out) – предикат, где out $\in$ OUT_{Sij}, показывающий готов ли результирующий параметр out экземпляра сервиса S_{ij} (т.е. успешно ли выполнился сервис и вернул ли какие-либо результаты).

Таким образом, возможна ситуация когда какой-либо экземпляр сервиса S_i зависит от экземпляра сервиса S_j, то есть S_i не может начать выполняться пока не завершится S_j.

Определим критерий выполнимости экземпляра S_{ia} сервиса S_i – экземпляр сервиса S_{ia} выполним тогда и только тогда, когда выполняется следующее условие – для любого элемента input $\in$ IN_{Sia}, одновременно принадлежащего какому-либо множеству из OUT_{Sj0} .. OUT_{Sjmax} (то есть, входной параметр одного сервиса является результатов работы другого сервиса) должно выполняться условие Ready(out), где out $\in$ OUT_{Sj} и input = out.

На основании описанной модели формулируется понятие зависимости сервисов по данным – два сервиса называются зависимыми по данным, если результирующие параметры одного сервиса являются входными параметрами для другого сервиса. Практическим применением данного понятия будет необходимость проверки готовности сервиса к выполнению – если хотя бы один из параметров сервиса еще не готов, то в данный момент сервис выполнять нельзя и следует вернуться к нему через какое-то время.

На программном уровне асинхронность достигается с помощью использования multi интерфейса библиотеки cURL [4], позволяющей инициировать HTTP-запросы к удаленным серверам в асинхронном режиме с установкой определенного callback'а, который срабатывает при завершении запроса.

Однако, в случае асинхронного выполнения сервисов возникает как необходимость проверки условия выполнимости сервиса, определенного в математической модели, так и реализация возможности получения результатов работы сервиса.

В силу сложности анализа непосредственного кода сценария на предмет зависимости между сервисами по данным, был реализован специальный объект класса ValueStore. Объекты класса ValueStore представляют собой контейнеры с заранее заданными идентификаторами, в которые складываются результаты работы сервисов при срабатывании callback функций, определенных для каждого асинхронно вызванного сервиса.

Если последовательность выполнения сервисов зависит от результата работы сервиса, то есть сценарий должен принять решение на основании результирующих данных какого-либо асинхронно выполняющегося сервиса, используется метод get(). Данный метод блокирует выполнение сценария до тех пор, пока данные не будут доступны, на основании которых можно будет принять решение.

Процесс асинхронного выполнения распределённых сервисов для рассмотренного выше сценария представлен на рисунке 2.

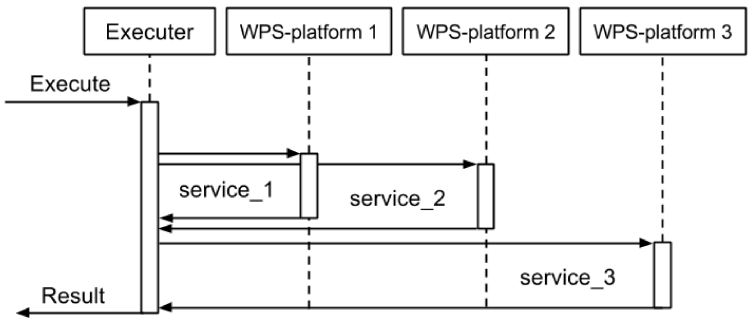

Рис. 2. Асинхронный вызов сервисов.

Апробация. Для апробации решения задачи ускорения выполнения сценариев распределенных сервисов с помощью организации асинхронного вызова сервисов был выбран пример расчета загрязнения от дорог на отдельно взятой области. В представленном сценарии вызывается три разных сервиса:

1. `roadToGrid` – преобразование векторного файла формата в SHP в растровый файл формата TIFF, примерное время выполнения на фиксированных расчетных данных – 50 секунд;
2. `numberOfDatasetItems` – получение числа элементов в векторном файле формата SHP, примерное время выполнения на фиксированных расчетных данных – 10 секунд;
3. `combineRaster` – сложение растровых файлов формата TIFF, примерное время выполнения на фиксированных расчетных данных – 30 секунд.

В качестве входных параметров сценарию передается массив векторных файлов формата SHP, в качестве результаты работы сценария ожидается созданный TIFF файл, в котором содержатся значения величин, взятых на основании элементов хранимых в SHP файле.

```
var input = [..];
function pollution_test(input) {
    var stores = ValueStore.factory(input.length);
    for (var i = 0; i < input.length; i++) {
        var localstore = new ValueStore();
        numberOfDatasetItems(input[i], {result: localstore})
        roadToGrid(input[i], localstore, {result: stores[i]});
    }
    if (ValueStore.areReady(stores) === true) {
        var finalstore = new ValueStore();
        combineRaster(stores, {result: finalstore});
        return finalstore.get();
    }
}
```

При синхронном вызове, то есть последовательном выполнении всех вызываемых сценариев в пределах одной нити выполнения общее время выполнения сценария равно 290 секундам, при асинхронном вызове сервисов общее время выполнения сценария равно 110 секундам.

Уменьшение времени выполнения сценария на 62 % достигнуто за счет сокращения выполнения цикла `for` внутри сценария – в случае синхронного вызова каждая итерация цикла занимает около 60 секунд, в то время как при асинхронном вызове одна итерация завершается примерно за 5 секунд – время, которое требуется на асинхронный запуск сервисов и возвращение в сценарий.

Результаты исследования. Результатом исследования, проведенного в данной работе, является организация асинхронного вызова сервисов в рамках технологии выполнения сценариев распределенных сервисов на языке программирования JavaScript, разработанной в тесной интеграции с Геопорталом ИДСТУ СО РАН. Данный подход оправдал себя как средство достижения цели – ускорения выполнения сценариев сервисов, что следует из проведенной апробации.

В рамках предложенной модели разработчик может реализовывать сценарий в привычном синхронном стиле программирования, построение последовательности и асинхронное выполнение сервисов производится автоматически на основе анализа зависимости по данным сценария.

Несмотря на то, что использование асинхронного вызова сервисов уже достаточно долгое время используется в аналогичных программных продуктах и технологиях, рассмот-

ренных в соответствующем разделе, и само по себе не ново, непосредственное использование асинхронности в JavaScript сценариях распределённых сервисов является новым и перспективным в силу сочетания следующих факторов: высокой скорости выполнения сценариев, большому набору управляющих конструкций языка написания сценариев, автоматической организации асинхронного выполнения сервисов и отслеживания зависимости выполняемых сервисов по данным.

Работа выполнена при финансовой поддержке РФФИ (грант № 14-07-00166(а))

ЛИТЕРАТУРА

[1] *И. В. Бычков, Г. М. Ружников, Р. К. Фёдоров, А. С. Шумилов*. Components of WPS environment for geoprocessing // Novosibirsk State University Journal of Information Technologies, Vol. 12, № 3, 2014.

[2] *J. de Jesus, P. Walker , M. Grant , S. Groom*, WPS orchestration using the Taverna workbench: The Science approach // Computers & Geosciences, Vol 47, 2012, p. 75-86.

[3] Электронный ресурс:

[4] Web Services Business Process Execution Language Version 2.0. **http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html** (дата обращения 24.06.2015).

[5] cURL Overview - multi interface overview. **http://curl.haxx.se/libcurl/c/libcurl-multi.html** (дата обращения 23.06.2015).