Е.О. Кабардинский, А.Г. Ивашко
Тюменский государственный университет, г.Тюмень
УДК 004.42

СРАВНИТЕЛЬНЫЙ АНАЛИЗ СЕРВИСНЫХ ШИН ПРЕДПРИЯТИЯ (ESB)

Аннотация. В статье представлена методика сравнения сервисных шин предприятия, а также результаты сравнения следующих ESB: Mule ESB, Apache Camel + Spring Boot

Ключевые слова: ESB, REST, SOAP, FTP, Web Services, Integration Tests, Mule ESB, Apache Camel, Spring Boot, Docker, Docker Composer

- **Сравнительный анализ сервисных шин данных (ESB)**

Интеграция информационных систем различного класса остается актуальной темой для многих компаний и государственных организаций, независимо от отрасли или сферы их деятельности [2]. При выполнении интеграции ключевым вопросом можно назвать выбор оптимальной платформы, которая позволит обеспечить надежность и быстродействие при совместной работе нескольких систем. В процессе выбора альтернативы, необходимо выделить преимущества одной из сервисных шин, исходя из требований к интеграционному решению и технологиям, используемым в интеграции. Для выбора наиболее подходящей ESB требуется выделить основные критерии оценки решения и провести анализ альтернатив на предмет соответствия.

Подходы к интеграции данных

Существует несколько основных подходов к интеграции данных в организациях [1]:

- Консолидация данных
- Федерализация данных
- Распространение данных
- Гибридный подход
- Сервисный подход

В соответствии с классификацией К. Диттриха [1], интеграцию данных возможно производить на нескольких уровнях архитектуры ИС организации:

Рис. 1 Классификация методов интеграции данных по Клаусу Диттриху

Сервисная шина данных позволяет реализовать интеграцию данных на следующих уровнях [1]:

- Common Data Storage (Общие системы хранения)
- Uniform Data Access (Унифицированный доступ к данным)
- Integration by Middleware (Интеграция средствами ПО промежуточного слоя)

Для выбора альтернативы среди различных сервисных шин предприятия, следует учесть технологии передачи данных, на которых основываются выше перечисленные методы интеграции [2]. Были выделены следующие транспортные технологии/протоколы:

- HTTP
- SOAP
- FTP

Учитывая данный список, было принято решение разработать программный комплекс, позволяющий провести автоматическое тестирование каждой из технологий, а также измерить вычислительную мощность и нагрузку. Каждый из автотестов должен быть реализован в виде web-сервиса.

- **Методология тестирования**

Основной задачей ESB является предоставление единого унифицированного доступа к данным предприятия по средствам web-сервисов [1]. Исходя из этого факта было принято решение о разработке клиентского приложения, задачей которого является вызов сервисов, предоставляемых ESB. Предполагается, что в одном тестовом окружении будет запущено несколько копий клиентского приложения, каждая из которых должна работать на отдельной физической машине. Главным требованием к клиентскому приложению является его способность последовательно вызывать определенный web-сервис в определенном временном промежутке. Для обеспечения синхронного вызова сервисов ESB все запущенные копии клиентских приложений должны управляться мастер-приложением (Рис. 2).

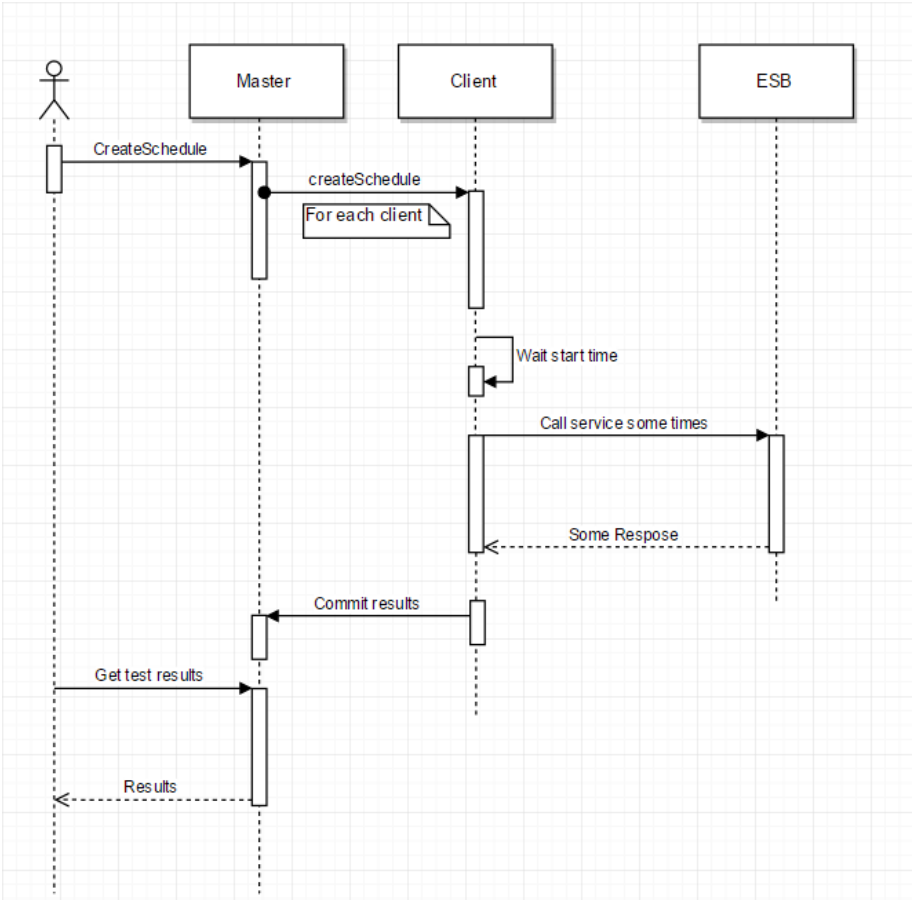

Рис. 2 Схема взаимодействия мастер-приложения, клиентских приложений, ESB

Так как ESB должна предоставлять доступ к сервисам предприятия [1], было принято решение разработать сервисы-заглушки, имитирующие реальные корпоративные системы:

- SOAP service
- RESTful service
- FTP server

Для разработки SOAP service и RESTful service была использована следующая XSD схема: «Purchase order schema for Example.Microsoft.com» [6].

Для замеров производительности и загрузки ESB были выбраны следующие параметры:

- Загрузка процессора
- Используемая память
- Длительность отклика на запрос клиента

Все параметры замеряются во время протекания автотеста. Загрузка процессора и используемая память замеряются на ESB машине, а длительность отклика - на клиентской машине.

Разработка окружения

Для быстрого развертывания сервисов-заглушек в произвольной компьютерной сети было решено использовать платформу контейнеризации приложений Docker [3]. Под контейнеризацией следует понимать управление псевдо-виртуальными машинами, а также виртуальными сетями. Данная платформа позволяет использовать публичные репозитории образов систем, что дает возможность использовать готовые, настроенные образы. Главной особенностью Docker является возможность расширять образы, используя встроенный декларативный язык создания образов [3]. Пример описания образа для SOAP service, использующего публичный образ Tomcat:

```
FROM tomcat:9.0
COPY tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml
```

```
COPY soap-service-1.0-SNAPSHOT.war
/usr/local/tomcat/webapps/soap-service.war
```

Листинг кода 1 Dockerfile для контейнера SOAP сервиса

Для управления развертыванием окружения было принято решение использовать Docker Composer tool. Пример описания сервисов-заглушек:

```
version: '2'
services:
  ftp:
    build: ftp/
    ports:
      - "21:21"
      - "30000-30009:30000-30009"
    environment:
      PUBLICHOST: 192.168.0.7
  rest:
    build: rest/
    ports:
      - "8080:8080"
  soap:
    build: soap/
    ports:
      - "8081:8081"
      - "8082:8080"
```

Листинг кода 2 docker-compose.yml файл-дескриптор тестового окружения

Была разработана следующая архитектура окружения (Рис. 3):

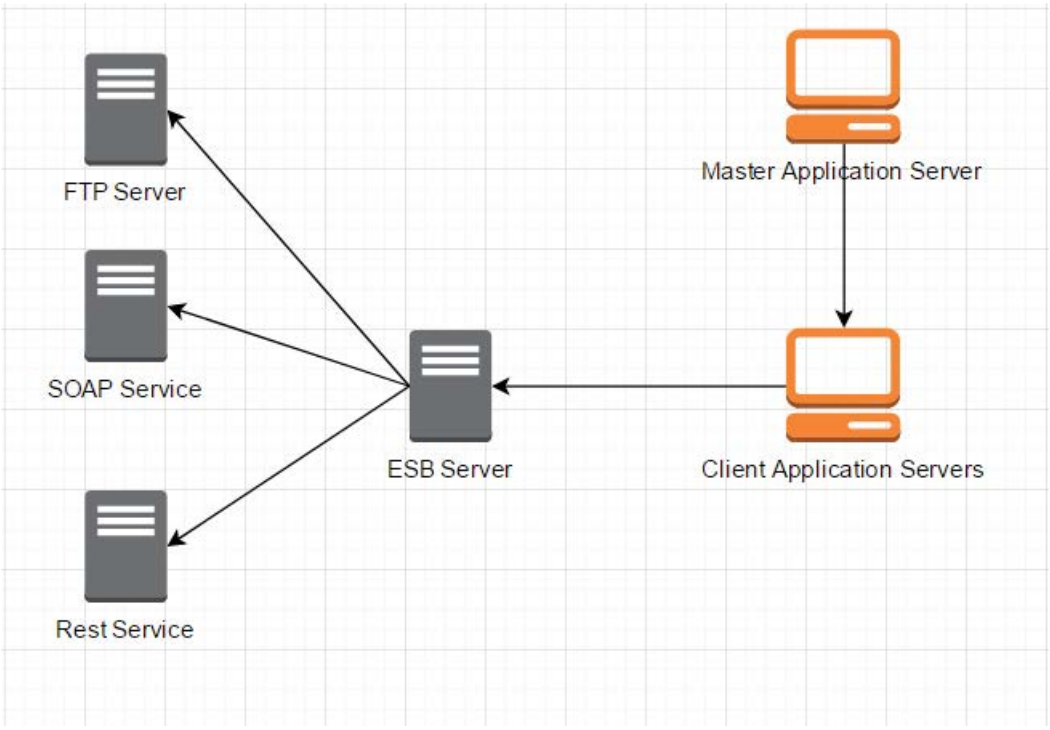

Рис. 3 Архитектура тестового окружения

Тестовое окружение предполагает использование одной физической машины для всех сервисов-заглушек, так как тестирование каждой из технологий происходит в отдельный промежуток времени. В данном тестировании использовалась следующая конфигурация тестового окружения:

- 1 физическая машина с ESB
- 2 физические машины с приложением-мастером, клиентом
- 1 физическая машина с сервисами-заглушками

Реализация ESB приложений

Для обеспечения унифицированного доступа к сервисам ESB был разработан набор интеграционных тестов, определяющих структуру запросов к сервисам, а также структуру ответов, что позволяет проверить ESB на предмет готовности к тестированию. Для реализации интеграционных тестов был использован инструмент Postman.

В данной статье были разработаны следующие тесты:

- SOAP Inbound – тестирование SOAP сервиса, реализованного в ESB

- SOAP Outbound – тестирование загрузки ESB при отправке SOAP запроса
- REST Inbound - тестирование REST сервиса, реализованного в ESB
- REST Outbound - тестирование загрузки ESB при отправке REST запроса
- FTP Outbound - тестирование загрузки ESB при отправке файла размером 5Mb по протоколу FTP

Пример реализации SOAP сервиса в Mule ESB представлен на рисунке 4.

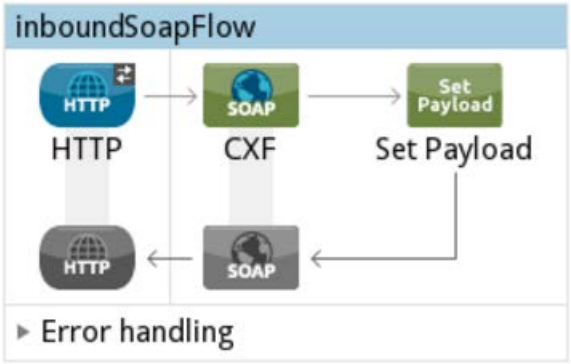

Рис. 4 Реализация SOAP сервиса в Mule ESB

- **Результаты**

В данной статье рассматривается тестирование двух ESB: Mule ESB и Apache Camel с web-сервером Spring Boot. Каждый из автотестов исполнялся в течение 5 минут. Полученные результаты представлены на рисунке 5.

	Outbound SOAP		Inbound SOAP		Outbound REST		Inbound REST		Outbound FTP	
Характеристика\ESB	Mule ESB	Apache Camel + Spring boot	Mule ESB	Apache Camel + Spring boot	Mule ESB	Apache Camel + Spring boot	Mule ESB	Apache Camel + Spring boot	Mule ESB	Apache Camel + Spring boot
Количество mb выделеных на 1 запрос	0,01969	0,02518	0,00577	0,01101	0,01209	0,01443	0,00506	0,00912	4,92155	0,34239
Среднее количество потребляемых mb	149,18	172,63	191,27	302,49	101,44	118,29	183,53	391,34	84,55	3,77
Средне отклонение потребленых mb	56,72	69,80	58,40	140,78	40,48	34,50	62,92	115,08	1,60	1,47
Максимальное потребление mb	233,99	310,92	282,79	541,10	146,93	184,97	280,90	509,61	98,43	7,53
Загрузка процессора в % на 1 запрос	0,0050	0,0065	0,0017	0,0014	0,0062	0,0048	0,0013	0,0010	1,3750	0,7761
Средний % загрузки процессора	7,0535	6,4567	8,2914	8,7699	5,4298	4,2290	5,9134	5,5303	0,3381	0,2045
Среднее отклонение % загрузки процессора	5,0927	4,8118	4,8773	4,4044	4,3718	3,4098	3,7736	3,1718	0,5932	0,3801
Максимальный % загрузка процессора	60,00	80,00	84,21	70,73	75,00	61,54	74,36	56,41	27,50	17,07
Среднее время отклика, мс	48,08	46,53	10,63	10,53	47,31	44,94	9,26	9,06	28 967,95	25 281,42
Среднее отклонение отклика, мс	19,74	13,56	6,77	6,23	20,59	15,32	6,25	5,38	6 124,48	2 014,54
Максимальный отклик, мс	3080	3070	3045	3021	3136	3134	3203	3046	59734	33984
Всего клиентских запросов	11885	12350	48975	49136	12156	12820	55475	55882	20	22
Замеров состояния ESB	2998	2999	3004	2998	2998	3000	2999	3001	2995	2995

Рис. 5 Результаты тестирования

Анализируя полученные результаты тестирования, были выявлены следующие особенности:

- Mule ESB потребляет меньшее количество оперативной памяти по сравнению с Apache Camel
- Apache Camel при обработке запросов загружает процессор меньше, чем MuleEsb
- Apache Camel выполняет более быструю обработку запросов клиентов

Разработанный программный комплекс распространяется под лицензией GNU GPL и доступен по адресу: https://github.com/ekabardinsky/esb-comparison

СПИСОК ЛИТЕРАТУРЫ

1. Dittrich, K.R. Data Integration - Problems, Approaches, and Perspectives / Patrick Ziegler and Klaus R. Dittrich // Database Technology Research Group Department of Informatics, University of Zurich. С. 4-5
2. Интеграция данных: синтаксис и семантика [Электронный ресурс]. Режим доступа: http://www.osp.ru/os/2009/10/11170978/ (Дата обращения: 15.04.2017)
3. Docker platform [Электронный ресурс]. Режим доступа: https://www.docker.com/ (Дата обращения: 15.04.2017)
4. Apache Camel [Электронный ресурс]. Режим доступа: http://camel.apache.org/ (Дата обращения: 15.04.2017)
5. Mule ESB | Enterprise Service Bus | Open Source ESB | MuleSoft [Электронный ресурс]. Режим доступа: https://www.mulesoft.com/platform/soa/mule-esb-open-source-esb (Дата обращения: 15.04.2017)
6. Sample XSD File: Purchase Order Schema [Электронный ресурс]. Режим доступа: https://msdn.microsoft.com/en-us/library/dd489284.aspx (Дата обращения: 15.04.2017)