МРНТИ 20.19.27 https://doi.org/10.58805/kazutb.v.1.22-218

NODE.JS И SCALA: ИНТЕГРАЦИЯ ДЛЯ СОЗДАНИЯ РАСПРЕДЕЛЕННЫХ МАСШТАБИРУЕМЫХ ПРИЛОЖЕНИЙ ДЛЯ АНАЛИЗА НАУЧНЫХ ТЕКСТОВ

А.С. Тургинбаева[1], С.А. Алтынбек[2*], У.К. Турусбекова[3], Н.Т. Азиева[4]

[1]Евразийский национальный университет им. Л. Н. Гумилева, Астана, Казахстан,

[2]Казахский университет технологии и бизнеса им.К.Кулажанова, Астана, Казахстан,

[3]Esil University, Астана, Казахстан,

[4]Евразийский национальный университет им. Л. Н. Гумилева, Астана, Казахстан,

e-mail: serik_aa@bk.ru

Node.js и Scala представляют собой две популярные технологии разработки, которые обладают уникальными особенностями и применяются в различных сферах. Node.js - серверный JavaScript, который стал основой для множества веб-приложений и сервисов. Scala, с другой стороны, является мощным языком программирования, который комбинирует преимущества функционального и объектно-ориентированного программирования. В этой статье мы сравниваем эти две технологии, рассматриваем их особенности, преимущества и недостатки, основные характеристики каждой технологии, сферы их применения. Node.js или Scala, оба эти инструмента предоставляют широкие возможности для создания мощных, масштабируемых и эффективных приложений и в статье рассматриваются возможности их интеграции, с целью создания масштабируемого приложения.

Ключевые слова: язык программирования Scala, Node.JS, научный текст, большие данные, неструктурированные данные, обработка данных, Apache Spark, распределенные вычисления, математический аппарат

NODE.JS AND SCALA: INTEGRATION FOR BUILDING DISTRIBUTED SCALABLE APPLICATIONS FOR SCIENTIFIC TEXT ANALYSIS

A.S. Turginbayeva[1], S.A. Altynbek[2*], U.K. Turusbekova[3], N.T. Aziyeva[4]

[1]L.N. Gumilyov Eurasian National University, Astana, Republic of Kazakhstan,

[2] Kazakh University of Technology and Business named after K. Kulazhanov,

Astana, Kazakhstan,

[3]Esil University, Astana, Republic of Kazakhstan,

[4]L.N. Gumilyov Eurasian National University, Astana, Republic of Kazakhstan,

e-mail: serik_aa@bk.ru

Node.js and Scala are two popular development technologies that have unique features and applications. Node.js is a server-side JavaScript that has become the foundation for many web applications and services. Scala, on the other hand, is a powerful programming language that combines the benefits of functional and object-oriented programming. In this article, we compare these two technologies, review their features, advantages and disadvantages, the main characteristics of each technology, and their areas of application. Node.js or Scala, both these tools provide great opportunities to create powerful, scalable and efficient applications and this article discusses how they can be integrated to create a scalable application.

Keywords: Scala programming language, Node.JS, scientific text, big data, unstructured data, data processing, Apache Spark, distributed computing, mathematical apparatus

NODE.JS ЖӘНЕ SCALA: ҒЫЛЫМИ МӘТІНДЕРДІ ТАЛДАЙТЫН

ТАРАТЫЛҒАН МАСШТАБТАЛАТЫН ҚОСЫМШАЛАРДЫ ҚҰРУҒА АРНАЛҒАН ИНТЕГРАЦИЯЛАУ

А.С. Тургинбаева[1], С.А. Алтынбек[2*], У.К. Турусбекова[3], Н.Т. Азиева[4]

[1]Л.Н. Гумилев атындағы Еуразия ұлттық университеті, Астана, Қазақстан,

[2] Қ. Құлажанов атындағы Қазақ технология және бизнес университеті АҚ, Астана, Қазақстан,

[3]Esil University, Астана, Қазақстан,

[4]Л.Н. Гумилев атындағы Еуразия ұлттық университеті, Астана, Қазақстан,

e-mail: serik_aa@bk.ru

Node.js және Scala бірегей ерекшеліктерге ие және түрлі салаларда қолданылатын екі танымал технологиядан тұрады. Node.js - көптеген веб-бағдарламалар мен сервистерге негіз болған серверлік JavaScript. Scala, екінші жағынан, функционалдық және объектілік-бағдарланған бағдарламалаудың артықшылықтарын біріктіретін қуатты бағдарламалау тілі болып табылады. Бұл мақалада біз осы екі технологияны салыстырамыз, олардың ерекшеліктерін, артықшылықтары мен кемшіліктерін, әрбір технологияның негізгі сипаттамаларын, оларды қолдану салаларын қарастырамыз. Node.js немесе Scala, бұл құралдардың екеуі де қуатты, ауқымды және тиімді қосымшаларды жасау үшін кең мүмкіндіктер береді және мақалада масштабталатын қосымшаны жасау мақсатында оларды біріктіру мүмкіндіктері қарастырылады.

Түйін сөздер: Scala бағдарламалау тілі, Node.JS, ғылыми мәтін, үлкен деректер, құрылымдалмаған деректер, деректерді өңдеу, Apache Spark, бөлінген есептеулер, математикалық аппараттар.

Введение. Node.js и Scala - это две популярные технологии разработки, с каждой из которых связаны уникальные возможности и характеристики. Node.js является серверной средой выполнения JavaScript, которая широко используется для создания веб-приложений и микросервисов. С другой стороны, Scala - это мощный и многогранный язык программирования, который сочетает в себе принципы функционального и объектно-ориентированного программирования. В данной статье мы предлагаем сравнительный анализ Node.js и Scala, рассматривая их преимущества и недостатки, области применения и примеры успешных масштабируемых проектов. Мы проведем обзор соответствующих сообществ разработчиков, экосистем инструментов и ресурсов для обучения, которые помогут вам выбрать наиболее подходящую технологию для вашего проекта [1].

Материалы и методы. Для начала сделаем краткий обзор этих языков программирования и сделаем анализ возможен ли их синтез для создания масштабируемых приложений.

Node.js - это серверная среда выполнения JavaScript, которая позволяет разработчикам создавать высокопроизводительные и масштабируемые веб-приложения. Основное назначение Node.js заключается в быстрой и эффективной обработке множества одновременных запросов, что делает его идеальным для создания веб-серверов и микросервисов. Вот некоторые преимущества использования Node.js:

1. Быстрота и масштабируемость: Node.js работает на основе событийного цикла и неблокирующего ввода-вывода, что позволяет эффективно обрабатывать множество запросов одновременно без блокировки потоков. Это делает Node.js идеальным для создания серверов с высокой производительностью и способностью масштабироваться.

2. Единый язык программирования: Node.js основан на JavaScript, что обеспечивает единый язык программирования как для фронтенда, так и для бэкенда. Это позволяет разработчикам легко переиспользовать код, обмениваться функциями и модулями между сторонами клиента и сервером.

3. Большое сообщество и экосистема инструментов: Node.js имеет огромное активное сообщество разработчиков, которое поддерживает множество пакетов и модулей, доступных через менеджера пакетов npm. Это обеспечивает широкие возможности для разработки и интеграции сторонних инструментов и библиотек.

4. Удобство разработки: Node.js предлагает простую и интуитивно понятную модель разработки на основе событий и коллбэков, что делает код более читабельным и легким в поддержке. Также существует множество инструментов разработки, таких как отладчики и среды разработки, которые значительно упрощают процесс создания приложений.

5. Возможность использования JavaScript-библиотек и фреймворков: Node.js позволяет разработчикам использовать популярные фреймворки и библиотеки JavaScript, такие как Express.js, Koa.js и Socket.io, для ускорения и упрощения разработки приложений.

Использование Node.js позволяет разработчикам создавать высокопроизводительные и эффективные веб-приложения с помощью одного языка программирования, широких возможностей и обширного сообщества разработчиков [2-3].

Scala (от "scalable language" - масштабируемый язык) - это мультипарадигменный язык программирования, разработанный для общего назначения. Он сочетает в себе функциональное и объектно-ориентированное программирование и предоставляет выразительный и гибкий синтаксис. Назначение и преимущества Scala:

1. Разработка приложений: Scala подходит для разработки различных типов приложений, от маленьких скриптов до сложных систем.

2. Расширение языка Java: Scala взаимодействует с Java и может использовать существующий код и библиотеки на Java.

3. Масштабируемость: Scala позволяет создавать масштабируемые приложения, которые могут эффективно обрабатывать большие объемы данных.

4. Выразительность: Scala имеет выразительный синтаксис, позволяющий писать компактный и понятный код. Он также поддерживает функциональное программирование, что делает его более гибким и мощным.

5. Интеграция с Java: Scala полностью совместим с Java, что позволяет использовать существующие Java-библиотеки и инструменты.

6. Поддержка конкурентного программирования: Scala предоставляет удобные инструменты для работы с конкурентностью, такие как акторы (actors) и параллельные коллекции.

7. Активное сообщество: Scala имеет активное и поддерживающее сообщество разработчиков, что обеспечивает доступ к ресурсам, библиотекам и поддержке.

В целом, Scala является мощным языком программирования, который обеспечивает удобство использования и гибкость при разработке больших и сложных приложений [5-6].

Scala и Node.js являются двумя разными языками программирования и имеют разные области применения. Однако, существуют сценарии, где интеграция между этими двумя языками может быть полезной. Одним из возможных сценариев использования Scala и Node.js вместе является создание микросервисной архитектуры. Scala может использоваться для разработки высокоэффективных и распределенных сервисов, в то время как Node.js может использоваться для разработки легковесных и высокопроизводительных веб-серверов, которые обрабатывают клиентский HTTP-трафик и выполняют сбор данных из различных сервисов. Также можно использовать Node.js в качестве прокси-сервера или шлюза, который общается с клиентами через HTTP и передает запросы к Scala-сервисам. В этом случае Node.js может играть роль точки входа и обработки исходящего трафика, а Scala-сервисы осуществляют более сложные операции, такие как обработка бизнес-логики и работа с базами данных. Еще один сценарий использования - это создание приложений, использующих библиотеки на Scala для вычислительных задач, а Node.js для создания пользовательского интерфейса и взаимодействия с клиентами. Например, вы можете использовать Scala для обработки и анализа больших объемов данных, а затем использовать Node.js для визуализации этих данных в виде интерактивного веб-интерфейса. Оба этих языка предлагают широкие возможности для разработки современных и высокопроизводительных приложений [7].

Представим, что мы разрабатываем масштабируемое приложение для анализа научных текстов. Для этого нам понадобятся высокопроизводительные вычисления, которые мы будем выполнять с помощью Scala, и веб-интерфейс, который будет разработан с использованием Node.js.

Пример интеграции Scala и Node.js для данного случая может выглядеть следующим образом:

- Node.js-сервер будет использовать HTTP-клиент, такой как Axios или Request, для отправки запросов к Scala-сервису с данными научных текстов и получения результатов анализа.

- Разработанный Scala-сервис может предоставлять API-методы, которые будут обрабатывать входные данные, Scala-сервис, который будет принимать научные тексты и проводить различные операции над ними, например, выполнение анализа частоты слов, извлечение ключевых фраз или классификацию текстов по тематике. Для этого мы можем использовать библиотеки, такие как Apache Spark, Natural Language Processing (NLP) или Machine Learning (ML) библиотеки на Scala. После заверше-

ния обработки текста параллельными алгоритмами и выполнения вычислении и возвращать результаты в формате JSON или другом удобном формате для Node.js-сервера.

Таким образом, веб-интерфейс на Node.js будет взаимодействовать с Scala-сервисами, отправлять им запросы для анализа научных текстов и получать результаты обратно, которые затем будут отображаться в пользовательском интерфейсе приложения. Этот пример демонстрирует интеграцию Scala и Node.js для создания масштабируемого приложения для анализа научных текстов, где Scala используется для выполнения высокопроизводительных операций над текстами, а Node.js - для разработки веб-интерфейса и взаимодействия с пользователем.

Результаты и обсуждения. Продемонстрируем пример передачи файла с Node.js на обработку в Scala, где мы используем простой HTTP запрос между двумя серверами.

1. На стороне Node.js вы можете использовать популярный модуль ‘axios‘ для выполнения HTTP запросов. Установите его, выполнив следующую команду в вашем проекте Node.js:

```
npm install axios
```

2. Файл .docx, который нужно отправить на обработку в Scala мы можем отправить, используя метод ‘axios.post‘ для отправки HTTP запроса на сервер Scala. Пример кода на Node.js может быть следующим:

JavaScript

```
const axios = require('axios');
const fs = require('fs');
// Чтение .docx файла
const fileData = fs.readFileSync('path/to/file.docx');
// Определение конфигурации запроса
const config = {
  method: 'post',
  url: 'http://scala-server:port/endpoint', // Замените 'scala-server'
  //и 'port' на адрес и порт вашего Scala сервера
  headers: {
    'Content-Type': 'application/octet-stream'
  },
  data: fileData
};
// Отправка запроса на сервер Scala
axios(config)
  .then(function (response) {
    // Обработка успешного ответа от сервера Scala
    console.log(response.data);
  })
  .catch(function (error) {
    // Обработка ошибки
    console.log(error);
  });
```

3. На стороне Scala сервера мы должны настроить обработку приходящего HTTP запроса. Вам понадобится использовать фреймворк, такой как Akka HTTP или Play Framework, чтобы обрабатывать приходящие запросы и извлекать содержимое файла .docx.

Пример кода на Scala с использованием Akka HTTP:

Scala

```
import akka.http.scaladsl.server.Directives._
import akka.http.scaladsl.model.StatusCodes
```

```
import akka.http.scaladsl.server.Route

val route: Route =
  path("endpoint") { // Endpoint, на который отправляется файл .docx
    post {
      entity(as[Array[Byte]]) { fileData =>
        // Обработка содержимого файла .docx

        complete(StatusCodes.OK) // Отправка успешного ответа
      }
    }
  }

// Запуск сервера на порту
val bindingFuture = Http().newServerAt("localhost", 8080).bind(route)
```

Пример выше демонстрирует создание простого "endpoint", который будет получать POST запрос с содержимым .docx файла. Обратите внимание, что нужно будет настроить адрес и порт, на котором работает Scala сервер, а также реализовать обработку содержимого файла .docx в вашем Scala коде, основываясь на используемых библиотеках для работы с .docx файлами, таких как Apache POI или Docx4j [8].

Как вы заметили выше для настройки входящего HTTP запроса мы используем Akka HTTP - это фреймворк для разработки серверных и клиентских HTTP-приложений на языке Scala или Java, использующий акторную модель, предоставляемую фреймворком Akka.

Выбор этого фреймворка обусловлен его некоторыми особенностями и возможностями:

1. Акторная модель: Akka HTTP полностью основан на акторной модели, что обеспечивает высокую пропускную способность, масштабируемость и отказоустойчивость.

2. Встроенная поддержка маршрутизации: Akka HTTP предоставляет мощный DSL для создания маршрутов, что делает его простым и гибким при построении REST API. Вы можете определить маршруты с помощью функций обработчиков или указать пути на основе URL.

3. Поддержка разных протоколов и методов: Akka HTTP поддерживает различные протоколы, такие как HTTP, HTTPS и WebSockets, а также поддерживает методы HTTP, такие как GET, POST, PUT и DELETE.

4. Асинхронная и неблокирующая обработка запросов: Akka HTTP использует асинхронную обработку запросов, что позволяет эффективно обрабатывать большое количество запросов с минимальным использованием ресурсов.

5. Легковесность и высокая производительность: Akka HTTP предлагает высокую производительность благодаря асинхронной обработке и использованию нескольких потоков.

6. Интеграция с другими компонентами Akka: Akka HTTP хорошо интегрируется с другими компонентами фреймворка Akka, такими как Akka Streams и Akka Actors, для создания сложных систем.

7. Расширяемость: Фреймворк Akka HTTP предоставляет возможность расширять его функциональность с помощью различных плагинов и библиотек.

Таким образом, интеграция этих двух мощных чзыков программирования дает нам возможность создания больших масштабируемых веб приложений, а дальнейшая паралельная обработка научных текстов на Scala не только увеличивает быстроту вычислений, но и дает нам возможность для выполнения различных операций над научными текстами, используя ряд библиотек и инструментов. Ниже приведены несколько примеров:

1. Анализ частоты слов:

- Для подсчета частоты слов можно использовать библиотеку Apache Lucene, которая предоставляет инструменты для работы с текстовыми данными, включая подсчет TF-IDF (Term Frequency-Inverse Document Frequency).

- Для визуализации результатов можно использовать библиотеку Apache Spark или другие инструменты для анализа данных.

2. Извлечение ключевых фраз:

- Для извлечения ключевых фраз можно исполь-

зовать библиотеку OpenNLP или Stanford NLP. Эти библиотеки предоставляют инструменты для обработки естественного языка, включая извлечение именованных сущностей и ключевых фраз.

- Для извлечения ключевых фраз можно также использовать алгоритмы машинного обучения, например, модели Word2Vec или TF-IDF.

3. Классификация текстов по тематике:

- Для классификации текстов по тематике можно использовать библиотеку Apache Spark ML, которая предоставляет алгоритмы классификации, включая Naive Bayes, SVM и Random Forest.

- Другой вариант - использовать библиотеку Deeplearning4j, которая предоставляет возможность обучать нейронные сети для классификации текстов.

Выводы. Node.js и Scala могут быть использованы для анализа научных текстов, каждый со своими особенностями и инструментами. С одной стороны, Node.js предоставляет мощные инструменты для обработки текстовых данных и анализа текстовых корпусов. Также он может использоваться для доступа к базам данных и поисковым движкам, что позволяет производить поиск и извлечение информации из больших наборов научных текстов. А Scala предоставляет возможности использования библиотек для обработки естественного языка, такие как Stanford CoreNLP, Breeze и Scalaj-HTTP. Он также предоставляет мощные средства для анализа текста, включая разбор синтаксиса, выделение слов, определение частей речи, распознавание именованных сущностей, инструменты для статистического анализа текстовых данных, включая моделирование тем, кластеризацию и классификацию текстов. А инструмент Scalaj-HTTP может использоваться нами для получения и обработки данных из внешних источников, таких как базы данных и веб-сайты научных журналов и т.д. В статье приведен пример такой интеграции Node.js и Scala, т.е использование микросервисной архитектуры, где Node.js и Scala могут работать по отдельности, но обмениваться данными и взаимодействовать друг с другом через API. Как видно такой подход позволяет использовать каждый инструмент по его сильным сторонам: Node.js для разработки веб-сервера и обработки запросов, а Scala для анализа научных текстов. При этом, использование REST API обеспечивает гибкость и расширяемость системы, позволяя добавлять дополнительные функции или инструменты анализа текста по мере необходимости.

Научно-исследовательская работа выполняется в рамках ГФ Министерством науки и высшего образования Республики Казахстан AR19677733 по теме «Разработка интеллектуальной распределенной системы параллельного анализа научных текстов» на 2023-2025 гг.

Литература

1.Boranbayev A., Shuitenov G., BoranbayevS. The Method of Analysis of Data from Social Networks Using Rapidminer //Advances in Intelligent Systems and Computing.- 2020.- 1229 AISC.- pp. 667-673.

2.Кантелон М. Node.js в действии / М. Кантелон. - СПб: Питер.- 2015. - 810 с.

3.Сухов, К. Node.js. Путеводитель по технологии / К. Сухов. . М.: ДМК Пресс.- 2015.- 416 с.

4.Одерски М., Спун Л., Веннерс Б. Scala. Профессиональное программирование = Programming in Scala: Updated for Scala 2.12.-СПб: Питер.- 2018.-688 с.-ISBN 978-5-496-02951-3.

5.Прокопец А. Конкурентное программирование на SCALA. -М.: ДМК Пресс.- 2017.-342 с. - ISBN 978-5-97060-572-1.

6.Хостманн, К. Функциональное программирование. SCALA для нетерпеливых /К. Хостманн. - М.: ДМК.- 2015.- 408 с.

7.Dean Wampler, Alex Payne. Programming Scala: Scalability = Functional Programming + Objects - 1st.- O'Reilly Media, 2009. - 448 p.- ISBN 0-596-15595-6.

8.Применяем Apache POI, docx4j и springframework.jdbc - https://habr.com/ru/articles/214435/. - Дата обращения: 27.09.2023

References

1.Boranbayev A., Shuitenov G., Boranbayev S. The Method of Analysis of Data from Social Networks Using Rapidminer //Advances in Intelligent Systems and Computing.- 2020.- 1229 AISC.- pp. 667-673.

2. Kantelon M. Node.js v dejstvii / M. Kantelon.- SPb: Piter.- 2015. - 810 s.

3.Suhov, K. Node.js. Putevoditel'po tehnologii / K. Suhov. -M.: DMK Press.- 2015.- 416 s.

4.Oderski M., Spun L., Venners B. Scala. Professional'noe programmirovanie - Programming in Scala: Updated for Scala 2.12.-SPb: Piter.- 2018.-688 s.-ISBN 978-5-496-02951-3

5.Prokopec A. Konkurentnoe programmirovanie na SCALA. -M.: DMK Press.- 2017.-342 s. - ISBN 978-5-97060-572-1.

6.Hostmann, K. Funkcional'noe programmirovanie. SCALA dlja neterpelivyh /K. Hostmann. - M.: DMK.- 2015.- 408 s.

7.Dean Wampler, Alex Payne. Programming Scala: Scalability = Functional Programming + Objects - 1st.- O'Reilly Media, 2009. - 448 p.- ISBN 0-596-15595-6.

8.Primenjaem Apache POI, docx4j и springframework.jdbc - https://habr.com/ru/articles/214435/. - Date of application: 27.09.2023

Сведения об авторах

Тургинбаева А.С. - магистр, Евразийский национальный университет им. Л. Н. Гумилева, Астана, Казахстан, e-mail: Tasheart@mail.ru

Алтынбек С.А. - PhD, Казахский университет технологии и бизнеса им.К.Кулажанова,Астана, Казахстан, e-mail: serik_aa@bk.ru;

Турусбекова У.К. - PhD, и.о. доцента, Esil University, Астана, Казахстан, e-mail: umut.t@mail.ru;

Азиева Нургул Танирбергеновна – старший преподаватель, кафедра «Информационная безопасность», Евразийский национальный университет им. Л. Н. Гумилева, Республика Казахстан, г. Астана, e-mail: nurgul_aziyeva@mail.ru

Information about the authors

Turginbayeva A.S. - Master, Eurasian National University. L. N. Gumileva, Astana, Kazakhstan, e-mail: Tasheart@mail.ru

Altynbek S.A. - PhD, Kazakh University of Technology and Business named after K.Kulazhanov, Astana, Kazakhstan, e-mail: serik_aa@bk.ru;

Turusbekova U.K. - PhD, Acting Associate Professor, ", Esil University, Astana, Kazakhstan, e-mail: umut.t@mail.ru;

Aziyeva Nurgul Tanirbergenovna – senior lecturer, Department of "Information Security", L.N. Gumilyov Eurasian National University, Republic of Kazakhstan, Astana, e-mail: nurgul_aziyeva@mail.ru