Bulletin of L.N. Gumilyov ENU. Mathematics. Computer science. Mechanics series, 2022, Vol. 140, №3, P.23-32
http://bulmathmc.enu.kz, E-mail: vest_math@enu.kz

МРНТИ: **20.23.21**

А.Б. Мусина [1] , С.С. Аубакиров [1] , П. Триго [2]

[1] *Казахский национальный университет имени аль-Фараби, аль-Фараби 71, Алматы, Казахстан*

[2] *Высший инженерный институт Лиссабона, ул. Консельейру Эмидиу Наварро, 1, 1959-007, Лиссабон, Португалия*

(E-mail: mussina.aigerim95@gmail.com, aubakirov.sanzhar@gmail.com, paulo.trigo@gmail.com)

Архитектура для непрерывного извлечения знаний из социальных сетей

Аннотация: Социальные сети уже давно играют неотъемлемую роль в повседневной жизни людей. Вся наша жизнь в реальном мире фиксируется и в цифровом пространстве. Социальные медиа и взаимодействующие с ними сети стали местом огромных возможностей для анализа данных. Их влияние на повседневную жизнь охватывает такие разные области, как цифровой маркетинг, анализ общественного мнения, мониторинг политической ситуации и уведомления о стихийных бедствиях. Любая задача обработки такого большого потока данных нуждается в целостной архитектуре, которая будет соответствовать анализируемому ресурсу. В представленной работе мы поставили перед собой задачу создать высоконагруженную, отказоустойчивую, масштабируемую систему для извлечения и обработки данных из различных социальных сетей и анализа данных в реальном времени. Решением выступает архитектура в виде комплекса модулей. Модули имеют свои особенности в зависимости от выполняемой работы, от сбора текстовых данных до непосредственной обработки и извлечения знаний.

Ключевые слова: Высоконагруженные системы, отказоустойчивость, масштабируемая архитектура, краулинг данных, Telegram.

DOI: https://doi.org/10.32523/2616-7182/bulmathenu.2022/3.3

2000 Mathematics Subject Classification: 68M01

Введение. Социальные сети в основном бесплатны и, следовательно, представляют собой мощную и широко распространенную инфраструктуру для общения с большой аудиторией. Например, компания может быстро получить обратную связь об определенном бренде и провести анализ влияния своих публикаций в сети [1]. Различные социальные сети в Интернете (OSN - Online Social Network) устанавливают и укрепляют свои собственные отношения между потребителями и производителями [2].

Цель нашего исследования - создать высоконагруженную, отказоустойчивую, масштабируемую архитектуру для массового извлечения и обработки, в режиме реального времени, данных из социальных сетей. В качестве первого примера мы решили взять набирающий популярность во всем мире мессенджер Telegram. На конец 2022 года в данной сети было около 350 000 активных пользователей из Казахстана [3]. В 2022 году ежемесячные активные пользователи Telegram во всем мире достигли 700 миллионов пользователей. Несмотря на быстрый рост, Telegram является относительно молодой OSN и еще не стала предметом интенсивных исследований. Авторы Telegram OSN предоставляют разработчикам множество общедоступных инструментов для создания собственных клиентских приложений [4].

Социальные сети генерируют огромное количество постоянно увеличивающихся данных, поэтому предъявляют строгие требования к любой архитектуре извлечения данных на основе поисковых роботов. Основные требования включают в себя:

1. Собирать информацию по разным социальным сетям в Интернете. Поскольку сканер социальных сетей должен работать с разными OSN, внешние API будут другими. Однако внутренняя обработка данных будет для всех одинакова.
2. Подключение новой OSN. Этот процесс должен быть простым.
3. Масштабируемость в отношении предварительной обработки и анализа данных.

Чтобы наша архитектура удовлетворяла указанным выше требованиям, она должна иметь следующие свойства.

Эластичность. Это степень, в которой система способна адаптироваться к изменениям в рабочей нагрузке, предоставляя и удаляя ресурсы в автономном режиме, так что в любой момент времени доступные ресурсы максимально приближены к текущему спросу [5].

Масштабируемость. Его можно разделить на «структурную масштабируемость и масштабируемость нагрузки». Структурная масштабируемость - это способность системы расширяться в выбранном измерении без значительных изменений в ее архитектуре. Масштабируемость нагрузки - это способность системы правильно работать при увеличении предлагаемого трафика [6].

Самостоятельное развертывание. Понимается как часть топологии развертывания приложения для реализации конкретной технической единицы [7]. Чаще всего под единицей развертывания понимается «стандартный контейнер». Цель стандартного контейнера - инкапсулировать программный компонент и все его зависимости в самоописывающем и переносимом формате, чтобы любая совместимая среда выполнения могла запускать его без дополнительных зависимостей, независимо от базовой машины и содержимого контейнера. Это определение из Open Container Initiative (OCI), которое объясняется в 5 принципах стандартных контейнеров [8].

В зависимости от цели анализа данных, могут меняться и требования по сбору данных. В качестве примера в данной работе мы рассмотрим задачу обнаружения событий, event detection. Данная задача применяется для различных OSN. Кроме того, «событие» имеет разные определения в зависимости от контекста и применения. Событие можно взять как «случай, вызвавший изменение объема текстовых данных, обсуждающих определенную тему в определенное время. Это событие характеризуется темой и временем и часто связано с такими объектами, как люди и местоположение» [9]. Далее на примере задачи обнаружения событий мы покажем работу нашей системы, построенной как комплекс модулей.

1. Обзор литературы. Архитектуры для краулинга данных с Интернет ресурсов помимо сбора данных также включают в себя определенную обработку данных. Ниже представлены работы последних лет с акцентом на архитектуру для Интернет краулинга. В последние годы все еще актуальны исследования на архитектуры для сбора данных с веб-сайтов [10–12]. Краулинг социальных сетей на данный момент возможен для различных сетей благодаря предоставленным API. В работе [13] предложена архитектура, которая собирает и анализирует данные по запросу пользователя относительно временного промежутка, локации и ключевых слов в нескольких сетях Google Trends, Twitter и Facebook [13]. Однако система не собирает данные постоянно в режиме реального времени.

Можно выделить два направления краулинга — общий [11] и сфокусированный [10]. Общий краулинг подразумевает сбор всей информации из интересующих ресурсов. Сфокусированный краулинг направлен на сбор информации по определенной тематике. Иногда это может решаться на уровне ресурсов с которых идет сбор данных, краулеру предлагаются для парсинга сайты с интересующей тематикой [14]. Однако если сайт располагает статьями на разные тематики, то чтобы не собирать всю информацию предлагаются архитектуры для сфокусированного краулинга [10]. В предложенной архитектуре имеется сервис фильтра по тематике. Перед тем как проиндексировать и внести URL ссылку в очередь для будущего краулинга, ресурс фильтруется по тематике при

помощи Deep Neural Network классификатора. В работе рассматриваются веб-страницы, которые в среднем больше чем сообщения в социальных сетях. Применение данной архитектуры для социальных сетей не представляется возможной.

Краулинг может концентрироваться не только на тематике текста, но и на пользователях [15]. Параллельно со сбором данных в представленной архитектуре присутствует модуль машинного обучения для непрерывного обучения модели по классификации пользователей, как тех чьи посты стоят внимания и не стоят. В работе представлена архитектура, где все модули связаны через Catenae, библиотекой Python для разработки масштабируемых графов по обработке потоков данных. Catenae использует для обмена сообщениями инструмент Kafka. Данный инструмент также используется в работе [16], для распределения сообщений с командами для краулинга. Хотя Kafka превосходит RabbitMQ по размеру обрабатываемых данных, но в микросервисной архитектуре со сложной маршрутизацией больше подходит RabbitMQ. Маршрутизация заключается в прохождении сообщений между краулерами и разными обработчиками данных. В данной работе представлена микросервисная архитектура для краулинга и обработки информации из социальных сетей в режиме реального времени. Подробно описаны примененные технологии.

2. Методология. В этом разделе мы описываем технологию, которая соответствует требованиям нашей архитектуры. В настоящее время существует парадигма разработки программного обеспечения, которая заключается в бизнес-процессах и архитектуре решения вышеуказанных проблем. Эта парадигма называется «Разработка облачных приложений» (Cloud-Native Application Development) [17]. Эта концепция относится к набору технологий и шаблонов проектирования, которые стали стандартом для создания крупномасштабных облачных приложений. Разработка программного обеспечения в этой парадигме обеспечивает свойства успешных облачных приложений, включая динамическую масштабируемость, максимальную отказоустойчивость, обновления без прерывания работы и безопасность. Чтобы сделать возможным создание приложений, отвечающих этим требованиям, мы описываем архитектуру микросервисов, которая играет центральную роль в облачном проектировании.

В огромном обзоре было проанализировано, собрано и обобщено более 50 работ, связанных с разработкой облачных приложений [18]. В результате авторы определили термин «собственное облачное приложение» следующим образом, далее перевод: «Облачное приложение (CNA) - это распределенная, гибкая и масштабируемая система (микро) сервисов, которая изолирует состояние в минимальном количестве компонентов с отслеживанием состояния. Приложение и каждый отдельный модуль развертывания этого приложения разработаны в соответствии с шаблонами проектирования, ориентированными на облако, и работают на гибкой платформе самообслуживания.»

Было предложено называть такие приложения IDEAL, чтобы приложение было [Isolated state] изолированным, [Distributed] имело распределенную архитектуру, было [Elastic] гибким в смысле горизонтального масштабирования, управлялось с помощью [Automated] автоматизированных систем и его компоненты должны быть слабо связаны [19]. Создание облачных приложений в этой парадигме приводит к следующим результатам [20]:

- более быстрое предоставление программных решений заказчику
- отказоустойчивость
- автоматизация восстановления
- легкое и быстрое горизонтальное масштабирование приложений
- возможность обрабатывать огромное количество данных

В основе нашего подхода к дизайну лежал наш первый кейс OSN - Telegram. В настоящее время разработчики социальных сетей обычно предоставляют общедоступные инструменты или библиотеки для взаимодействия со своей системой и ее данными. Выше было сказано, что Telegram становится все более популярным. Мы решили сначала создать краулер для социальных сетей на основе Telegram OSN. Согласно официальной библиотеке

баз данных Telegram (TDLib), предоставленной Telegram, мы создали наше Client API приложение.

Комплекс модулей должен включать в себя следующие компоненты:

- сбор данных
- хранение данных
- обработка данных
  - токенизация
  - лемматизация
  - определение тематики текста
  - определение настроения текста
  - подсчет частотности токенов и лемм
  - обнаружение события

Модули обработки данных всегда одинаково связаны с хранилищами данных и модулями сбора данных. Однако их внутренняя логика может меняться в зависимости от конечной цели анализа социальных сетей.

3. Результаты и Обсуждение. В этом разделе мы описываем нашу архитектуру для извлечения и обработки данных из социальных сетей. Структура базы данных также представлена в разделе 3.3.

3.1 Облачная архитектура. Мы разработали облачную архитектуру на основе микросервисов. Микросервисы - это декомпозиция монолитных бизнес-систем на независимо развертываемые сервисы, которые выполняют одну задачу. Основной способ взаимодействия между сервисами в архитектуре облачных приложений - через опубликованные и версионные API (совместная работа на основе API). В нашей архитектуре микросервисы общаются через очередь сообщений.

Отдельные развертываемые блоки архитектуры спроектированы и взаимосвязаны в соответствии с набором облачных шаблонов, таких как приложение с двенадцатью факторами [21], Circuit Breaker [22]. Методология двенадцатифакторных приложений - это методология создания приложений «программное обеспечение как услуга». Прерыватель цепи используется в архитектуре микросервисов для предотвращения каскадных сбоев во время взаимодействия сервисов. Эти передовые методы предназначены для создания приложений, обеспечивающих переносимость и отказоустойчивость при развертывании в Интернете.

Мы используем гибкую платформу OpenStack, которая используется для развертывания и эксплуатации этих микросервисов через автономные единицы развертывания (контейнеры). Эта платформа предоставляет дополнительные операционные возможности поверх инфраструктуры IaaS, такие как автоматическое масштабирование экземпляров приложений и масштабирование по запросу, управление работоспособностью приложений, динамическая маршрутизация, балансировка нагрузки, а также агрегирование журналов и показателей.

На рис. 1 мы изобразили процесс сбора данных через нашу облачную архитектуру и общий вид архитектуры. Каждое Новое сообщение, созданное в OSN, обнаруживается и анализируется краулером. Во-первых, краулер сохраняет сообщение в базе данных. Во-вторых, он помещает текстовое содержимое сообщения в очередь. Сервис очередей имеет области обмена и определенные очереди. Каждый модуль обработки данных имеет свой консьюмер и получает свои данные через очередь. Эти микросервисные модули описаны ниже. После всех этапов обработки данных вся извлеченная информация попадает в базу данных.

3.2 Технологии. В разделе 2 мы рассмотрели парадигму IDEAL. Приложения созданные по данной парадигме явно соответсвуют основным требованиям для системы сбора и обработки данных. В нашей архитектуре мы использовали следующие технологии:

- Docker Swarm позволяет создать отказоустойчивую, масштабируемую, легко и быстро восстанавливающуюся систему. Автоматизация восстановления

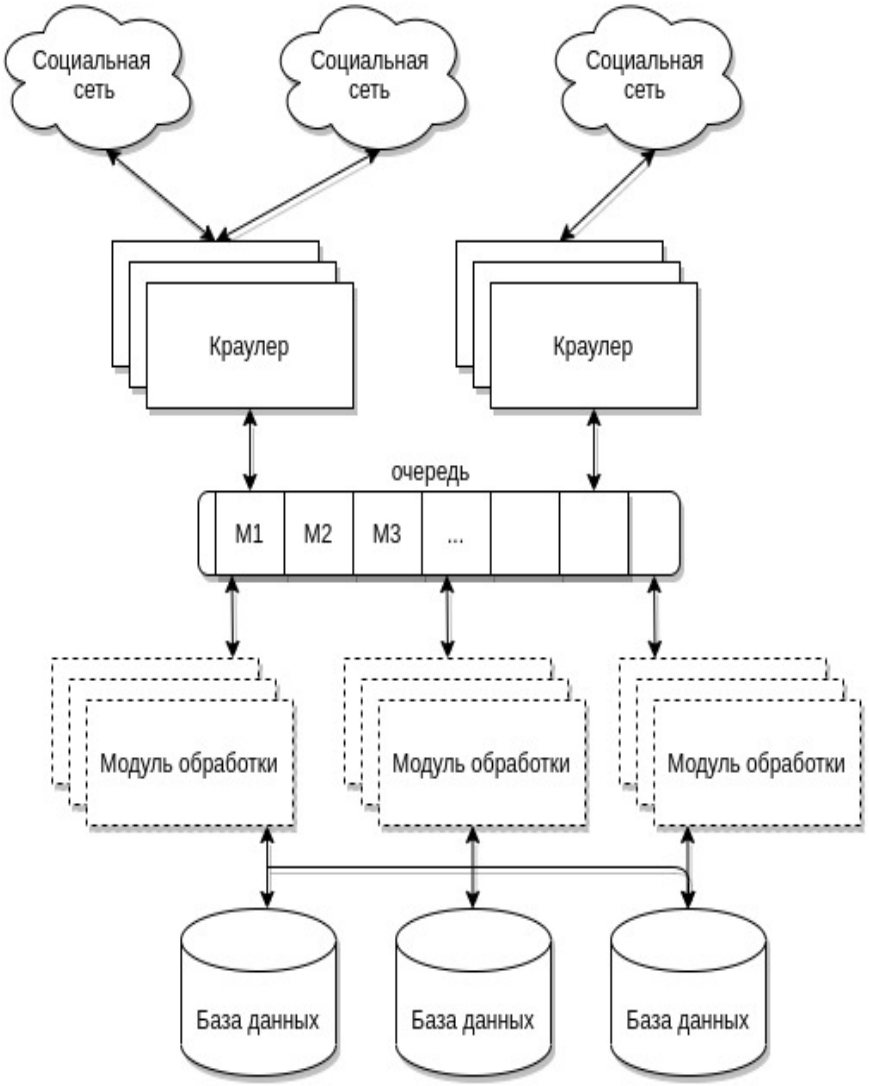

Рис. 1. Архитектура

контейнеров после сбоя, благодаря health checking, обеспечивает отказоустойчивость. Контейнеры дают возможность реализовать CI/CD, что ускоряет доставку кода в тестовую и производственную среду. Таким образом программные решения быстро предоставляются заказчику. За счёт функции репликации контейнеров достигается горизонтальное масштабирование приложений в соответствии с нагрузкой. На рис. 1 Crawler и Data processing представляют собой микросервисы, которые контролируются через Docker Swarm.

- RabbitMQ используется для создания очередей между Crawler и Data processing микросервисами. Очереди дают нам возможность обрабатывать огромное количество данных асинхронно, консистентно и отказоустойчиво.
- Apache Elasticsearch используется для быстрого доступа к данным. Процесс обнаружения событий в большом информационном пространстве будет легче представить как интерактивный процесс. Это значит, что мы хотим в дальнейшем предоставить пользователю возможность менять параметры для поиска. Elasticsearch является отличным решением для быстрого поиска данных.
- Traefik даёт возможность управления http трафиком внутри docker swarm ingress сети. Благодаря Traefik реализован вышеописанный паттерн Circuit Breaker.

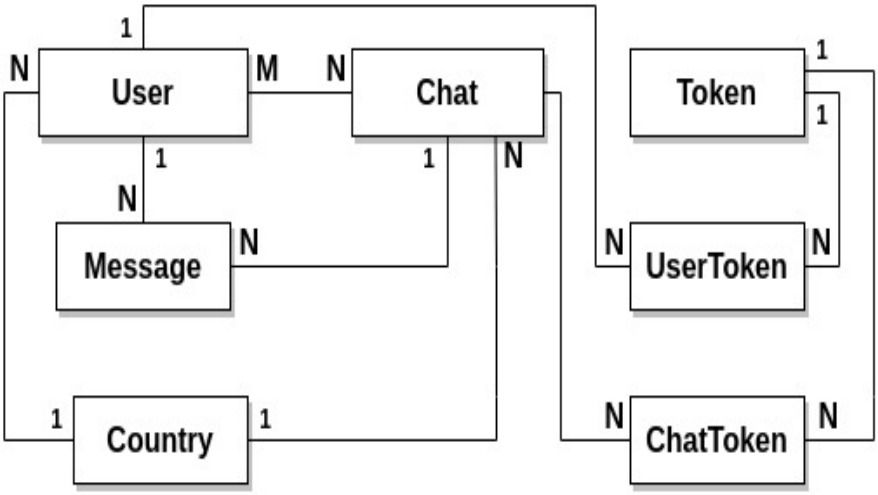

Рис. 2. Концептуальная модель данных

- Open Stack управляет ресурсами и виртуальными машинами, что сильно облегчает работу с серверным оборудованием.

3.3 Проектирование моделей данных. Открытый API от Telegram позволяет собирать много информации о пользователях, чатах и сообщениях. На рис. 2 показана концептуальная модель данных нашей реляционной базы данных.

Первым модулем предварительной обработки была токенизация текста сообщения [23]. Токен в нашем случае - это юниграмма. Перед токенизацией текст очищается от русских стоп-слов, взятых из RussianAnalyzer в библиотеке Apache Lucene [24]. Для каждого токена считается частота, с которой он появляется в сообщениях отдельно взятых пользователей и чатов. Например, в Казахстане во время пандемии COVID-19 и ЧП государство выплачивало 42 500 тенге компенсации людям, потерявшим работу или доход. Этот токен '42500' имеет общее количество более 11000 в 34 чатах. Однако в большей степени этот токен активно использовался только в одном новостном канале.

Вторым инструментом предварительной обработки была лемматизация сообщений на русском и английском языках. Для русскоязычных сообщений мы использовали библиотеку на базе Lucene [25].

Третий модуль представляет собой микросервис для извлечения топиков. Данный модуль может послужить основой для решения поставленной в пример задачи обаружения событий. На данный момент для извлечения топиков используется самый простой метод, вычисление tf-idf. Мы рассматриваем каждое сообщение как «документ», объединяем все сообщения из чатов за один день и вычисляем tf-idf для каждого токена. Топ-K токенов с самым высоким tf-idf берутся в качестве K топиков дня (мы используем $K = 5$). В отличие от других модулей, которые были написаны на языке программирования Java, данный модуль написан на языке программирования Python. Тем самым комплекс модулей демонстрирует независимость разработки модулей в работе над одной большой задачей.

Все модули в данной архитектуре легко масштабируются по мере надобности. Например, первые два модуля явно требуют большого количества реплик, чтоб не задерживать сообщения из OSN в очереди. Количество реплик третьего модуля зависит от частоты запросов со стороны пользователей системы. Независимость модулей позволяет настраивать их и их окружение самым оптимальным образом.

Мы собираем сообщения с 19.02.2020 от публичных групп и каналов из Казахстана. В начале апреля 2021 мы включили несколько чатов из России, Белоруссии, Украины и Узбекистана. На 05.11.2022 база данных насчитывает 12 000 чатов, 808 000 пользователей и 4 220 000 сообщений. Рост данных представлен на рис. 3.

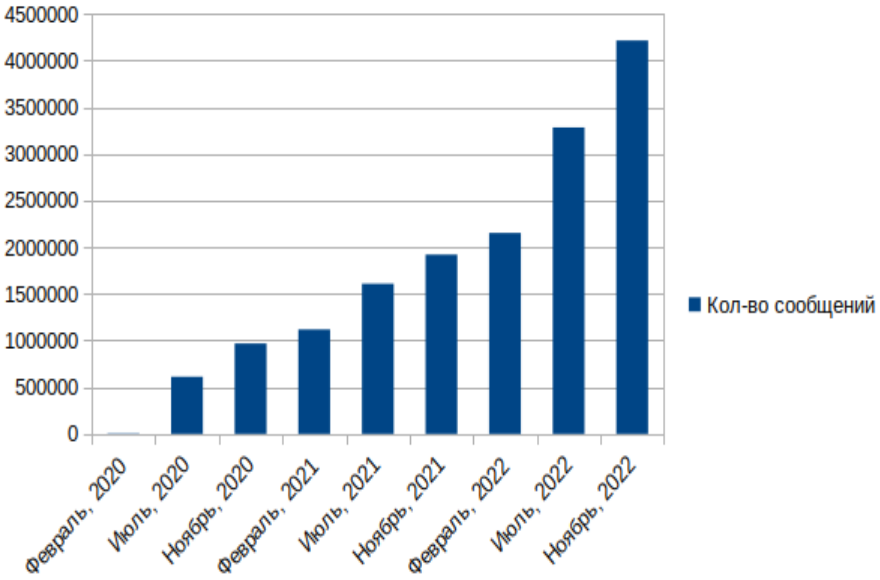

Рис. 3. Рост пополняемых данных

Полученная архитектура показала свою работоспособность. Комплекс модулей отвечает всем требованиям при сборе и обработке данных в режиме реального времени. Предствленное облачное решение может применяться для изучения гораздо более сложных задач по обработке и анализу данных из социальных сетей. Ознакомится более подробно с разработкой можно на нашей странице в GitHub [26].

Заключение. Мы разработали высоконагруженную, отказоустойчивую, масштабируемую архитектуру, состоящую из комплекса модулей, для извлечения и хранения данных из социальных медиа, используя парадигму разработки облачных приложений. Краулер собирает сообщения из 2765 чатов Telegram. Микросервис обработки данных выполнен в виде масштабируемых модулей, которые можно легко расширить с помощью дополнительных инструментов. Результаты данной работы мы планируем использовать в будущей работе. Мы собираемся провести дополнительный обзор литературы по тематике обнаружения событий в социальных сетях и определить точный и эффективный алгоритм обнаружения событий и их взаимосвязей.

Список литературы

1 Klepek M., Starzyczn H. Marketing communication model for social networks// Journal of Business Economics and Management 19, 2018. –N 3. –P. 500–520. DOI: https://doi.org/10.3846/jbem.2018.6582.
2 Sharma S., Verma H. V. Social Media Marketing: Evolution and Change // Social Media Marketing, 2018. –P. 19–36. DOI: https://doi.org/10.1007/978-981-10-5323-8_2.
3 Telegram channels Kazakhstan [Электронный ресурс] - URL: https://kaz.tgstat.com/en. (Дата обращения: 01.11.2021).
4 Telegram Database Library [Электронный ресурс] - URL: https://core.telegram.org/tdlib. (Дата обращения: 10.10.2021).
5 Herbst N., Kounev S., Reussner R. Elasticity in cloud computing: What it is, and what it is not // International Conference on Autonomic Computing, 2013. –P. 23-27.
6 Bondi A. B. Characteristics of Scalability and Their Impact on Performance // Proceedings of the second international workshop on Software and performance - WOSP '00, 2000. –P. 195-203. DOI: https://doi.org/10.1145/350391.350432.
7 Inzinger C., Nastic S., Sehic S., Vogler M., Li F., Dustdar S. MADCAT: A Methodology for Architecture and Deployment of Cloud Application Topologies // 2014 IEEE 8th International Symposium on Service Oriented System Engineering, 2014. DOI: https://doi.org/10.1109/sose.2014.9.
8 Open Container Initiative [Электронный ресурс] URL: https://opencontainers.org/. (Дата обращения: 15.01.2021).

9 Dou, W., Wang, X., Ribarsky, W., Zhou, M. Event detection in social media data // In Proceedings of the IEEE VisWeek Workshop on Interactive Visual Text Analytics-Task Driven Analytics of Social Media Content, January 2012. –P. 971–980.
10 Elaraby M., Mohamed Sh., Moftah H., Rashad M. A new architecture for improving focused crawling using deep neural network // Journal of Intelligent & Fuzzy Systems. –2019. –N. 1. –P. 1233-1245. DOI: https://doi.org/10.3233/JIFS-182683.
11 Elaraby M., Moftah H., Mohamed Sh., Rashad M. Elastic Web Crawler Service-Oriented Architecture Over Cloud Computing // Arabian Journal for Science and Engineering. –2018. –N. 12. –P. 8111–8126. DOI: https://doi.org/10.1007/s13369-018-3241-z.
12 Zhang J., Shan Yu., Peng F. Web-Crawling Architecture in Accounting and Finance Research // Journal of Computer Information Systems. –2022. –N. 5. –P. 875-887. DOI: https://doi.org/10.1080/08874417.2021.1931983.
13 Kalatzis N., Roussaki I., Matsoukas Ch., Paraskevopoulos M., Papavassiliou S., Tonoli S. Social Media and Google Trends in Support of Audience Analytics: Methodology and Architecture // In Proceedings of the DATA ANALYTICS 2018, November 2018. –P. 39-45.
14 Schedlbauer J., Raptis G., Ludwig B. Medical informatics labor market analysis using web crawling, web scraping, and text mining // International Journal of Medical Informatics. –2021. –V. 150. –P. 104453-104461. DOI: https://doi.org/10.1016/j.ijmedinf.2021.104453.
15 Martínez-Castaáo R., Losada D. E., Pichel J. C. Real-Time Focused Extraction of Social Media Users // IEEE Access. –2022. –V. 10. –P. 42607-42622. DOI: https://doi.org/10.1109/ACCESS.2022.3168977.
16 You Ch., Zhu D., Sun Yu., Ye A., Wu G., Cao N., Qiu J., Zhou H. SNES: Social-Network-Oriented Public Opinion Monitoring Platform Based on ElasticSearch // Computers, Materials & Continua. –2019. –N. 3. –P. 1271-1283. DOI: https://doi.org/10.32604/cmc.2019.06133.
17 Gannon D., Barga R., Sundaresan N. Cloud-Native Applications // IEEE Cloud Computing 4. –2017. –N. 5. –P. 16–21. DOI: https://doi.org/10.1109/mcc.2017.4250939.
18 Kratzke N., Quint P.-C. Understanding Cloud-Native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study // Journal of Systems and Software 126. –2017. –P. 1–16. DOI: https://doi.org/10.1016/j.jss.2017.01.001.
19 Fehling C., Leymann F., Retter R., Schupeck W., Arbitter P. Cloud Computing Patterns //Springer-Verlag Wien. –2014. DOI: https://doi.org/10.1007/978-3-7091-1568-8.
20 Stine Matt. Migrating to Cloud-Native Application Architectures / O'Reilly Media, Inc. 2015.
21 The twelve-factor app [Электронный ресурс] - URL: http://12factor.net. (Дата обращения: 15.01.2021).
22 Microservices - a definition of this new architectural term [Электронный ресурс] - URL: http://martinfowler.com/articles/microservices.html. (Дата обращения: 12.01.2021).
23 Aubakirov S.S., Trigo P., Ahmed-zaki D.Zh. Bulding a model to predict classifier accuracy // Eurasian Journal of Mathematical and Computer Applications 5. –2017. –N. 3. –P. 4–14. DOI: https://doi.org/10.32523/2306-3172-2017-5-3-4-14.
24 Lucene 4.0.0 analyzers-common api [Электронный ресурс] - URL: https://lucene.apache.org/core/4_0_0/analyzers-common/. (Дата обращения: 10.09.2021).
25 Russian morphology for apache lucene [Электронный ресурс] - URL: https://github.com/AKuznetsov/russianmorphology. (Дата обращения: 10.09.2021).
26 GitHub knowledge-extraction-system [Электронный ресурс] - URL: https://github.com/knowledge-extraction-system. (Дата обращения: 03.10.2021).

А.Б. Мусина [1] , С.С. Аубакиров [1] , П. Триго [2]

[1] *әл-Фараби атындағы Қазақ Ұлттық Университеті, Алматы, Қазақстан*
[2] *Лиссабон жоғары инженерлік институты, Консельйейру Эмидиу Наварро көш., 1, 1959-007, Лиссабон, Португалия*

Әлеуметтік желілерден үздіксіз білімді шығаруға арналған архитектура

Аннотация: Әлеуметтік желілер ұзақ уақыт бойы адамдардың күнделікті өмірінде ажырамас рөл атқарады. Нақты әлемде біздің бүкіл өміріміз цифрлық кеңістікте жазылған. Әлеуметтік медиа және онымен әрекеттесетін желілер деректерді талдау үшін тамаша мүмкіндіктер орнына айналды. Олардың күнделікті өмірге әсері сандық маркетинг, қоғамдық пікірді талдау, саяси мониторинг және апаттар туралы хабарландыру сияқты әртүрлі салаларды қамтиды. Осындай үлкен деректер ағынын өңдеудің кез келген тапсырмасы талданған ресурсқа сәйкес келетін когерентті архитектураны қажет етеді. Ұсынылған жұмыста біз әртүрлі әлеуметтік желілерден деректерді алу мен өңдеу және нақты уақыт режимінде деректерді талдау үшін жоғары жүктелген, ақауларға төзімді, масштабталатын жүйені құру міндетін қойдық. Шешім модульдер кешені түріндегі архитектура болып табылады. Модульдердің мәтіндік деректерді жинаудан бастап тікелей өңдеуге және білімді шығаруға дейінгі орындалатын жұмысқа байланысты өзіндік сипаттамалары бар.

Түйін сөздер: Жоғары жүктеме жүйелері, ақауларға төзімділік, масштабталатын архитектура, деректерді тексеру, Telegram

A.B. Mussina [1] , S.S. Aubakirov [1] , P. Trigo [2]

[1] *al-Farabi Kazakh National University, 71 al-Farabi Ave., Almaty, Kazakhstan*
[2] *Instituto Superior de Engenharia de Lisboa, R. Conselheiro Emidio Navarro 1, 1959-007, Lisbon, Portugal*

Architecture for enduring knowledge-extraction from online social networks

Abstract: Nowadays social networks and media play significant role in daily life. All our life in the real world is recorded in the digital space as well. Scientists have enormous potential in researching issues such as social influence on top news and top news influence on society. Its impact on daily life spans such diverse areas as digital marketing, public opinion analysis, political monitoring and disaster notification. Any task of processing such a large data stream needs a coherent architecture that will fit the analyzed resource. In the presented work, we set ourselves the task of creating a highly loaded, fault-tolerant, scalable system for extracting and processing data from various social networks and analyzing data in real time. The solution is architecture in the form of a set of modules. Modules have their own characteristics depending on the work performed, from collecting textual data to direct processing and extraction of knowledge.

Keywords: Highly loaded, fault-tolerant, scalable architecture, data crawling, Telegram.

References

1 Klepek M., Starzyczn H. Marketing communication model for social networks, Journal of Business Economics and Management 19, 2018. –N 3. –P. 500–520. DOI: https://doi.org/10.3846/jbem.2018.6582.
2 Sharma S., Verma H. V. Social Media Marketing: Evolution and Change, Social Media Marketing, 2018. P. 19–36. DOI: https://doi.org/10.1007/978-981-10-5323-8_2.
3 Telegram channels Kazakhstan [Electronic resource]. Available at: https://kaz.tgstat.com/en. (Accessed: 01.11.2021).
4 Telegram Database Library [Electronic resource]. Available at: https://core.telegram.org/tdlib. (Accessed: 10.10.2021).
5 Herbst N., Kounev S., Reussner R. Elasticity in cloud computing: What it is, and what it is not, International Conference on Autonomic Computing, 2013. –P. 23-27.
6 Bondi A. B. Characteristics of Scalability and Their Impact on Performance, Proceedings of the second international workshop on Software and performance - WOSP '00, 2000. P. 195-203. DOI: https://doi.org/10.1145/350391.350432.
7 Inzinger C., Nastic S., Sehic S., Vogler M., Li F., Dustdar S. MADCAT: A Methodology for Architecture and Deployment of Cloud Application Topologies, 2014 IEEE 8th International Symposium on Service Oriented System Engineering, 2014. DOI: https://doi.org/10.1109/sose.2014.9.
8 Open Container Initiative [Electronic resource]. Available at: https://opencontainers.org/. (Accessed: 15.01.2021).
9 Dou, W., Wang, X., Ribarsky, W., Zhou, M. Event detection in social media data, In Proceedings of the IEEE VisWeek Workshop on Interactive Visual Text Analytics-Task Driven Analytics of Social Media Content, January 2012. P. 971–980.
10 Elaraby M., Mohamed Sh., Moftah H., Rashad M. A new architecture for improving focused crawling using deep neural network, Journal of Intelligent & Fuzzy Systems. 2019. N. 1. P. 1233-1245. DOI: https://doi.org/10.3233/JIFS-182683.
11 Elaraby M., Moftah H., Mohamed Sh., Rashad M. Elastic Web Crawler Service-Oriented Architecture Over Cloud Computing, Arabian Journal for Science and Engineering. 2018. N. 12. P. 8111–8126. DOI: https://doi.org/10.1007/s13369-018-3241-z.
12 Zhang J., Shan Yu., Peng F. Web-Crawling Architecture in Accounting and Finance Research, Journal of Computer Information Systems. 2022. N. 5. P. 875-887. DOI: https://doi.org/10.1080/08874417.2021.1931983.
13 Kalatzis N., Roussaki I., Matsoukas Ch., Paraskevopoulos M., Papavassiliou S., Tonoli S. Social Media and Google Trends in Support of Audience Analytics: Methodology and Architecture, In Proceedings of the DATA ANALYTICS 2018, November 2018. P. 39-45.
14 Schedlbauer J., Raptis G., Ludwig B. Medical informatics labor market analysis using web crawling, web scraping, and text mining, International Journal of Medical Informatics. 2021. V. 150. P. 104453-104461. DOI: https://doi.org/10.1016/j.ijmedinf.2021.104453.
15 Martínez-Castaáo R., Losada D. E., Pichel J. C. Real-Time Focused Extraction of Social Media Users, IEEE Access. 2022. V. 10. P. 42607-42622. DOI: https://doi.org/10.1109/ACCESS.2022.3168977.
16 You Ch., Zhu D., Sun Yu., Ye A., Wu G., Cao N., Qiu J., Zhou H. SNES: Social-Network-Oriented Public Opinion Monitoring Platform Based on ElasticSearch, Computers, Materials & Continua. 2019. N. 3. P. 1271-1283. DOI: https://doi.org/10.32604/cmc.2019.06133.
17 Gannon D., Barga R., Sundaresan N. Cloud-Native Applications, IEEE Cloud Computing 4. 2017. N. 5. P. 16–21. DOI: https://doi.org/10.1109/mcc.2017.4250939.
18 Kratzke N., Quint P.-C. Understanding Cloud-Native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study, Journal of Systems and Software 126. 2017. P. 1–16. DOI: https://doi.org/10.1016/j.jss.2017.01.001.

19 Fehling C., Leymann F., Retter R., Schupeck W., Arbitter P. Cloud Computing Patterns, Springer-Verlag Wien. –2014. DOI: https://doi.org/10.1007/978-3-7091-1568-8.
20 Stine Matt. Migrating to Cloud-Native Application Architectures / O'Reilly Media, Inc. 2015.
21 The twelve-factor app [Electronic resource]. Available at: http://12factor.net. (Accessed: 15.01.2021).
22 Microservices - a definition of this new architectural term [Electronic resource]. Available at: http://martinfowler.com/articles/microservices.html. (Accessed: 12.01.2021).
23 Aubakirov S.S., Trigo P., Ahmed-zaki D.Zh. Bulding a model to predict classifier accuracy, Eurasian Journal of Mathematical and Computer Applications 5. 2017. №. 3. P. 4–14. DOI: https://doi.org/10.32523/2306-3172-2017-5-3-4-14.
24 Lucene 4.0.0 analyzers-common api [Electronic resource]. Available at: https://lucene.apache.org/core/4_0_0/analyzers-common/. (Accessed: 10.09.2021).
25 Russian morphology for apache lucene [Electronic resource]. Available at: https://github.com/AKuznetsov/russianmorphology. (Accessed: 10.09.2021).
26 GitHub knowledge-extraction-system [Electronic resource]. Available at: https://github.com/knowledge-extraction-system. (Accessed: 03.10.2021).

Сведения об авторах:

Мусина А.Б. – ***автор для корреспонденции,*** студентка PhD Казахского национального университета им. аль-Фараби, кафедра «Информатики», аль-Фараби 71, Алматы, Казахстан.

Аубакиров С.С. – PhD, Старший научный сотрудник в Казахском национальном университете им. аль-Фараби, аль-Фараби 71, Алматы, Казахстан.

Триго П. – PhD, адъюнкт-профессор Высшего инженерного института Лиссабона, ул. Консельейру Эмидиу Наварро, 1, 1959-007, Лиссабон, Португалия Лиссабон, Португалия.

Mussina A.B. – ***corresponding author,*** PhD student of the al-Farabi Kazakh National University, Department of Computer Science, 71 al-Farabi Ave., Almaty, Kazakhstan.

Aubakirov S.S. – PhD, Senior Researcher at the al-Farabi Kazakh National University, 71 al-Farabi Ave., Almaty, Kazakhstan.

Trigo P. – PhD, Adjunct Professor at Instituto Superior de Engenharia de Lisboa, R. Conselheiro Emidio Navarro 1, 1959-007, Lisbon, Portugal.

Поступила в редакцию 06.05.2022