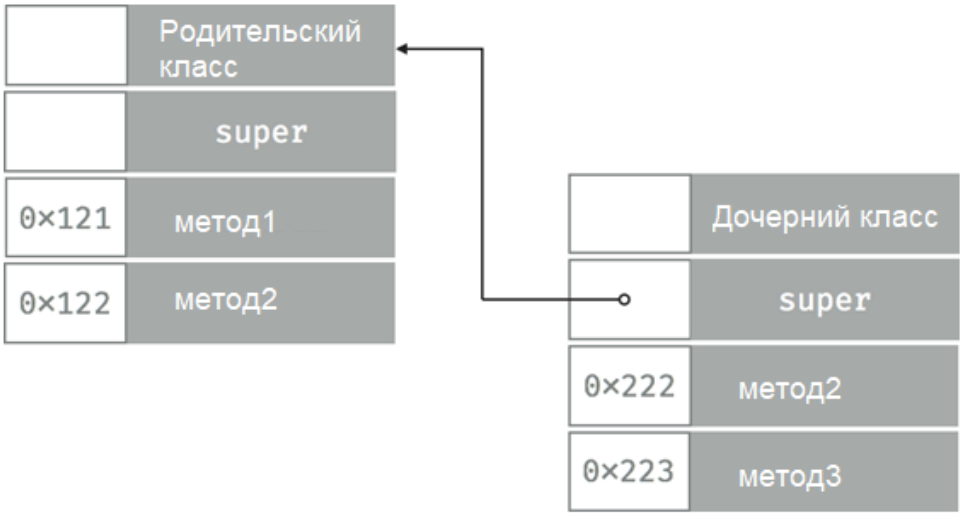

Рисунок 3 – Пример реализации механизма пересылки сообщений

Список использованных источников:

1. Understanding Virtual Tables in C++ [Электронный ресурс]. – Режим доступа: https://pabloariasal.github.io/2017/06/10/understanding-virtual-tables/ – Дата доступа: 19.03.2019.
2. The Black Magic of (Java) Method Dispatch [Электронный ресурс]. – Режим доступа: https://shipilev.net/blog/2015/black-magic-method-dispatch/ – Дата доступа: 19.03.2019.
3. Method Dispatch in Swift [Электронный ресурс]. – Режим доступа: https://www.raizlabs.com/dev/2016/12/swift-method-dispatch/ – Дата доступа: 20.03.2019.
4. Secrets of Swift's Speed [Электронный ресурс]. – Режим доступа: https://www.mikeash.com/pyblog/friday-qa-2014-07-04-secrets-of-swifts-speed.html – Дата доступа: 20.03.2019.
5. Interface dispatch [Электронный ресурс]. – Режим доступа: https://lukasatkinson.de/2018/interface-dispatch/ – Дата доступа: 20.03.2019.
6. Exploring Dynamic Dispatch in Rust [Электронный ресурс]. – Режим доступа: https://alschwalm.com/blog/static/2017/03/07/exploring-dynamic-dispatch-in-rust/ – Дата доступа: 21.03.2019.
7. Блэнди Дж., Орендорф Дж., Программирование на языке Rust / пер. с анг. А. А. Слинкина. – М.: ДМК, Пресс, 2018. – 550 с.: ил, стр. 227 – 228.
8. Method resolution order in Python Inheritance [Электронный ресурс]. – Режим доступа: https://www.geeksforgeeks.org/method-resolution-order-in-python-inheritance/ – Дата доступа: 21.03.2019.
9. Objective-C vs Swift messages dispatch [Электронный ресурс]. – Режим доступа: https://blog.untitledkingdom.com/objective-c-vs-swift-messages-dispatch-9d5b7fd58327 – Дата доступа: 21.03.2019.
10. Мартин Р., Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2018. – 352 с.: ил., стр. 60 – 63.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ СИСТЕМЫ ХРАНЕНИЯ ОБЪЕКТОВ АВТОРСКИХ ПРАВ НА ОСНОВЕ ТЕХНОЛОГИИ БЛОКЧЕЙН

Глоба А.А.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Ганжа В.А. – к.физ.-мат.н.,доцент

Разрабатываемая система служит для хранения объектов авторских прав с использованием технологии блокчейн. Блокчейн в общем виде – это выстроенная в соответствии с определенными правилами цепочка блоков, содержащих какую-либо информацию. Обычно копии цепочек хранятся распределённо на разных устройствах в пределах одной сети. С этой базы данных невозможно ничего удалить или провести замену/подмену блока, доступно только чтение.

В данный момент разрабатываемая БД представляет собой структуру, представленную на рис.1:

В ходе разработки системы для хранения авторских прав, было принято решение разделить БД на несколько цепочек, где каждому типу данных (текстовые, аудио-, изображения) соответствовала своя цепочка. Более того, на каждую отдельную цепочку накладывались свои правила для прохождения валидации. Такой подход обеспечивает гибкость и относительно легкую

расширяемость. Каждой цепочке соответствует свой файл с данными, в котором связь осуществляется через поле id, а контроль целостности – с помощью хэша MD5.

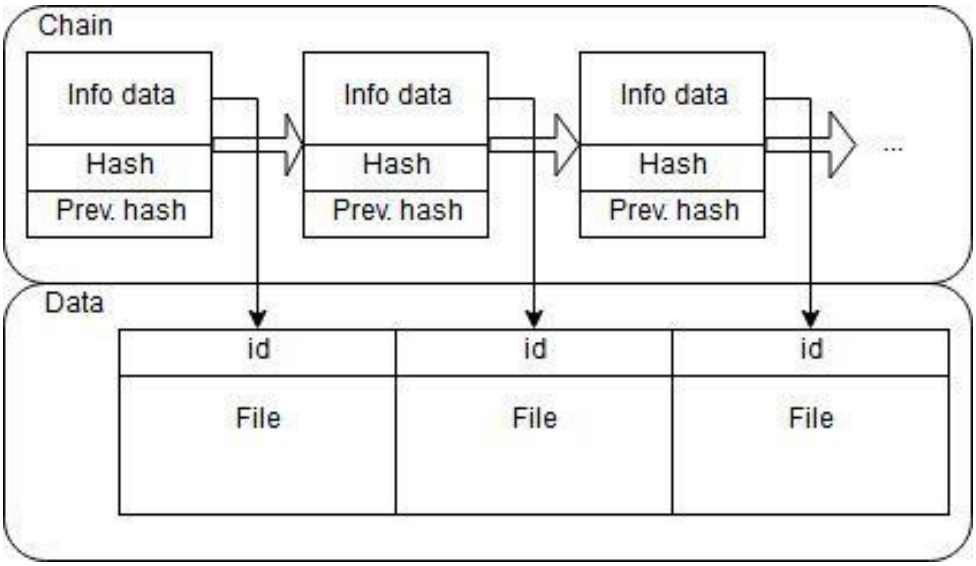

Рис.1 – структура данных в разрабатываемой системе

Сама разрабатываемая система разделена на 3 уровня:

1. Уровень данных (цепочки блокчейн).
2. Уровень API – HTTP-интерфейс для взаимодействия с цепочками.
3. Уровень представления – графический интерфейс пользователя.

Важным условием работоспособности разрабатываемой системы является возможность обмена данными и обновлениями между узлами сети, а также разрешение возможных конфликтов при этом. Для этого используется следующий алгоритм (пример для одного типа цепочки):

1. Узел запрашивает цепочки от всех известных ему узлов.
2. Если получает цепочку, длина которой больше, чем длина цепочки данного узла ($L_{new} > L_{current}$), то новая цепочка проходит валидацию. Так же при этом от удаленного узла скачивается блок данных с файлами, которые также проходят валидацию.
3. Если проверка новой цепочки и новых данных прошла успешно, то узел обновляет замещает свою цепочку новой, а также дополняет данные новыми файлами, если нет – продолжает поиск.
4. Поиск обновлений для каждого типа данных (текст, изображения и т.п.) проходит отдельно и независимо друг от друга.

Валидация при этом проходит в несколько этапов и считается успешной, если выполнены следующие условия:

1. Хэш-суммы блоков корректны.
2. Хэш-сумма блока N и значение поля previousHash блока N+1 совпадают для всех блоков.
3. Данные (текст, изображения, аудиозаписи) соответствуют описанным в системе требованиям, в том числе и на повтор/плагиат.
4. Оценка текстового файла на плагиат, в частности, совершается путём попарного сравнения нового и старых текстов с помощью функций TF-IDF и косинуса векторов.

TF-IDF — статистическая мера, используемая для оценки важности слова в контексте документа. Вес некоторого слова прямо пропорционален частоте употребления этого слова в строке и обратно пропорционален частоте употребления слова во всех строках коллекции.

TF-IDF состоит из:

1. **TF** (*term frequency* — частота слова) — соотношение количества вхождений некоторого слова к общей сумме числа слов документа. По формуле (1) оценивается важность слова в пределах отдельного документа, где в числителе находится число вхождений слова в документ, а в знаменателе — общее число слов в данном документе.

$$tf(t,d) = \frac{n_t}{\sum_k n_k}, \quad (1)$$

где n_t – число вхождений слова t в документ.

2. **IDF** (*inverse document frequency* — обратная частота документа) — инверсия частоты, с которой некоторое слово встречается в документах коллекции.

$$idf(t, D) = \frac{|D|}{|\{t \in d_i\}|}, \quad (2)$$

где |D| - количество документов в коллекции, $|\{t \in d_i\}|$ – число документов из коллекции D, в которых встречается слово t.

Как результат выполнения функции TF-IDF получаем связный список типа «ключ-значение», где каждому слову присвоен определенный коэффициент. Полученные значения используются для определения сходства текстов с помощью косинус-функции: $cos(a, b) = \frac{(\underline{a},\underline{b})}{|\underline{a}|\times|\underline{b}|}$ (3). Полученное значение и определяет сходство текстов, если оно слишком велико, новый файл не проходит валидацию.

Аналогичные проверки на плагиат планируется реализовать и для остальных типов данных.

Список использованных источников:

1. Jones K. S., A statistical interpretation of term specificity and its application in retrieval.
2. Saul Schleimer, Daniel S. Wilkerson, Alex Aiken Winnowing: Local Algorithms for Document Fingerprinting.

ОБРАБОТКА ПОСТОЯННОГО ПОТОКА ДАННЫХ С АППАРАТНЫХ И ВИРТУАЛЬНЫХ ПЛАТФОРМ CISCO

Губин И.Ю.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Иванов Н.Н. – к.т.н.,доцент

В настоящее время потребности абонентов к пропускной способности сети растут с каждым днём, так стараясь удовлетворить эти потребности, операторы связи ведут активное внедрение новый аппаратных и виртуальных решений, которые могли бы обеспечить обработку большого объема трафика. А при большом количестве оборудования, как аппаратного, так и виртуального, за ним надо наблюдать, получать статистические данные, обрабатывать их и строить прогнозы для лучшей балансировки и реакции на проблемы в моменты пиковой нагрузки. Все это вытекает в сбор, обработку и хранение огромного количество данных.

В ходе доклада будет рассказано, каким образом и какие данные собираются, как с аппаратных, так и виртуальных платформ. Как происходит предобработка и обработка статистических данных оборудования. Как хранятся эти данные и каким образом получается к ним доступ.

Так же будет изложено о пользе сбора и анализа данных, какие выводы можно сделать на основе полученных отчетов, как это может повлиять на коммерческую составляющую, а также как статистические данные помогают избежать аварий и вовремя среагировать при достижения определенных лимитов по ресурсам и много другое.

Будут рассмотрены варианты визуализации данных, посредством Grafana – открытой платформы для анализа и мониторинга, на которой отображаются нужные графики с использованием обработанных данных из баз данных.

Кроме вариантов визуализации, так же будут рассмотрены алгоритмы получения данных напрямую с аппаратной платформы, а также сбор данных после предобработки специализированным ПО.

Будет затронута тема обработки данных на языке программирования Python 3, а также библиотека, предназначенная для работы с большими потоками и объемами данных pandas – это программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня. Предоставляет специальные структуры данных и операции для манипулирования числовыми таблицами и временными рядами.

В итоге будут сделаны выводы о проделанной работе, полученном опыте при разработке и полезности в применении данного подхода для продакшн проекта. Будет рассказано о плане будущего развития проекта, какие модули будут добавлены по возможности в будущем и простоте расширения.