МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ имени ГУМАРБЕКА ДАУКЕЕВА»
Кафедра IT-инжиниринг

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой
PhD, профессор
_____ А.А. Досжанова
«____» _____ 2020 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

На тему: Интеграция методов сериализации и кластеризации данных для обработки и передачи больших объемов информации

Специальность 6М070400 – «Вычислительная техника и программное обеспечение»

Выполнил магистрант группы МВТн-18-1 _____ Баймурат Д.Р.

Научный руководитель д.т.н., профессор _____ Ахметов Б.С.
«____» _____ 2020 г.

Консультанты по применению
вычислительной техники: ст. преп. _____ Ж.С. Айткулов
«____» _____ 2020 г.

Нормоконтролер: PhD, ст. преп. _____Б.Р. Абсатарова
«____» _____ 2020 г.

Рецензент: PhD, ассистент-профессор _____ Н.К. Мукажанов
«____» _____ 2020 г.

Алматы 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ имени ГУМАРБЕКА ДАУКЕЕВА»
Институт систем управления и информационных технологий
Кафедра IT-инжиниринг

Специальность 6М070400 – «Вычислительная техника и программное обеспечение»

ЗАДАНИЕ
на выполнение магистерской диссертации

Магистранту Баймурат Данияр Рзадинулы

Тема проекта: Интеграция методов сериализации и кластеризации данных для обработки и передачи больших объемов информации

Утверждена приказом по университету № 155 от «23» октября 2019 г.

Срок сдачи законченного проекта «_1_» _____июня_____ 2020 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): Техническая документация по Java, Python, операционная система Windows 7,8,10.

Перечень вопросов, подлежащих разработке в диссертационной работе, или краткое содержание диссертационной работы:

а) обзорно-аналитическая часть;
б) проектирование системы умного дома;
в) разработка модели обучения;

Перечень графического материала (с точным указанием обязательных чертежей): представлены 13 таблиц, 83 иллюстрации.

Основная рекомендуемая литература:

1 Саммерфилд М. Программирование на Python 3. Подробное руководство, Издательство Символ-Плюс, 2009. – 608 с.

2 Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino, издательство БХМ-Петербург, 2012. – 256 с.

3 Гудфеллов Я., Бенджио И., Курвиль А. Глубокое обучение, Издательство ДМК-Пресс, 2018. – 652с.

Консультации по проекту с указанием относящихся к ним разделов работы

Раздел	Консультант	Сроки	Подпись
Программная часть	Рамазанова А.М.	05.05 – 02.05.20	
Нормоконтролер	Абсатарова Б.Р.	26.04 – 29.05.20	

График
подготовки диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Аналитическая часть работы	31.01.20	
Проектирование системы умного дома	02.03.20	
Разработка модели обучения	17.04.20	

Дата выдачи задания «25» __октября_____ 2020 г.

Заведующий кафедрой _____ А.А. Досжанова

Научный руководитель проекта _____ Б.С. Ахметов

Задание принял к исполнению магистрант _____ Д.Р. Баймурат

АННОТАЦИЯ

Предметы научно-исследовательской работы: интеграции методов сериализации и кластеризации для обработки больших объемов информации используя протокол simply binary encoding (SBE) и параллельно сравнивая другие методы. Проанализированы методы сериализации и кластеризации данных. Изучены сфера интеграции данной работы. Приводится сравнение протокола SBE с другими протоколами Protobuf, JSON. Объектом диссертационного исследования является последствия, возникающая в мире информационных технологии с генерацией больших данных каждый день.

АҢДАТПА

Зерттеу жұмысының тақырыптары: қарапайым екілік кодтау (SBE) протоколын қолдана отырып, үлкен көлемде ақпаратты өңдеуге арналған сериалдау және кластерлеу әдістерін біріктіру және параллельді басқа әдістерді салыстыру. Деректерді сериялау және кластерлеу әдістері талданады. Бұл жұмыстың интеграциялану аясы зерттелген. SBE протоколы басқа Protobuf, JSON протоколдарымен салыстырылады. Диссертациялық зерттеудің объектісі - ақпараттық технологиялар әлемінде күн сайын үлкен мәліметтер пайда болатын салдарлар.

ANNOTATION

Subjects of research work: integrating serialization and clustering methods for processing large amounts of information using the simple binary encoding (SBE) protocol and comparing other methods in parallel. The methods of serialization and clustering of data are analyzed. The scope of integration of this work is studied. The SBE protocol is compared with other Protobuf, JSON protocols. The object of the dissertation research is the consequences that arise in the world of information technology with the generation of big data every day.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Главным приоритетом данной научной работы является разработка приложения для передачи больших объемов финансовой информации ускоренно, используя протокол SBE. В настоящее время, в век инноваций, информационные технологии играют важную роль во всех сферах деятельности. Объемы данных передаваемой и получаемой информации показывает быстрый рост и появляется проблема необходимости увеличения скорости передачи данных. В соответствии с этим, в данной работе практическая значимость заключается в том, что разработанное приложение предоставляет возможность ускоренно передавать большой объем данных с уменьшением задержек в получении информации. В большинстве районов нашей страны распространение информационных технологий довольно высоко. Фондовая биржа является одной из этих областей. Как известно, фондовый рынок является высокотехнологичной сферой в наше время. И на основе этих прогрессов, создается несколько торговых платформ, брокерских систем, которые справляются с большой нагрузкой высокоскоростных каналов связи. Из этого следует, что востребованность приложений, которые обеспечивают огромную компрессию данных для передачи значительных объемов информации очень высока не только в сфере финансовой системы, а так же в маркетинге, бизнесе, телекоммуникации, торговле, логистике и государственных управлениях. Согласно статистике, весь объем хранимых и получаемых данных растет каждый год и проблема быстрой трансмиссии данных очень важна и имеет важную роль в XXI веке. Научная новизна работы заключается в разработке метода для ускоренной передачи больших объемов финансовых материалов с задержками, сниженными на минимум, который позволяет декодированные и десериализованные данные сохранять в базу данных и представлять их в виде биржевых японских свеч. Для выделения качественных отличий протокола SBE был проведен сравнительный анализ с JSON с указанием преимуществ и недостатков.

1 ИССЛЕДОВАНИЕ ТЕХНОЛОГИЙ И МЕТОДОВ ПЕРЕДАЧИ, А ТАКЖЕ ОБРАБОТКА БОЛЬШИХ ДАННЫХ

1.1 Исследование и анализ больших данных

1.1.1 Большие данные

Данные - это цифры, знаки или символы, на которых выполняются операции с помощью компьютера, хранящиеся в виде электрических сигналов, передаваемые и записываемые на магнитных, оптических или механических носителях записи [1].

Большие данные - это совокупность данных из различных источников, полученных от конкретного определения до слабо определенных данных, полученных из таких источников, как человек или машина [2].

Большие данные - это развивающийся термин, характеризующий большой объем структурированных, полуструктурных и неструктурных данных, который может быть произведен для информации и использован в проектах машинного обучения и других передовых аналитических приложениях [3].

Большие данные часто описываются под «3V»: «volume» - экстремальный объем данных, «variety» - многообразие данных и «velocity» - скорость обработки данных. Эти характеристики впервые Gartner определили в отчете, опубликованном в 2001 году, аналитик ведущей в мире исследовательской и консультационной компании Gartner Дуг Лейни. С недавних времен к большому числу данных можно отнести подлинность «veracity», «value» - ценность, и добавлены еще несколько» V характеристик, таких как изменчивость «variability». По крайней мере большие данные не приравниваются к какому -либо конкретному объему данных, этот термин часто используется для описания отпечатанных терабайтов , петабайтов и даже эксабайтов со временем накопления данных[4]. Такие объемные данные могут поступать из многих различных источников, таких как система бизнес-транзакций, база данных клиентов, медицинские записи, журналы интернет-трафика, мобильные приложения, социальные сети, собранные результаты научных экспериментов, машинные данные и IoT датчики данных, используемые в реальном времени в среде IoT (интернет-предметы). Использование средств интеллектуального анализа данных или программных средств для их подготовки перед анализом данных или оставлением данных в необработанном виде может быть предварительно обработано с использованием программного обеспечения [4].

Большие данные включают также структурированные данные в базах данных и хранилищах данных SQL, неструктурированные данные, такие как файлы документов, хранящиеся в кластерах Hadoop и частично структурированные данные, такие как поточная передача данных из журналов веб-сервера или датчиков. Кроме того, большие данные включают в себя несколько источников данных одновременно, которые не могут быть

объединены по-разному.

Мы создаем цифровую модель мира уже последние 15 лет. Мир окутан сетями, люди используют миллиарды мобильных телефонов. Огромное количество датчиков развивает интернет. И мы на планете Земля практический создаем цифровую модель реального мира.

Каждый день мы генерируем 2.5 млрд тб. И представить в ближайшее время, сколько у нас будет накопленных данных очень трудно и антиутопично. И нужно использовать все это огромное многообразие данных для того чтобы получить выгоду, будь то это получение эффективности или дохода.

Данные в XXI веке такой же природный ресурс и важный природный ресурс, как нефть в XX. И те, которые смогут выгодно и эффективно использовать эти зетабайты данных и сможет вынести пользу - получить преимущество. Как это сделать? Что делать с этими данными? Как их хранить? И самое главное, как их передавать?

Например, проект анализа больших данных содержит данные о прошлых продажах, данные о возврате и данные о покупателе для этого продукта, сравнением данных онлайн-анализа можно попытаться измерить прибыль продукта и будущие продажи.

Скорость передачи данных с большими сроками относится к скорости, которая генерируется и должна быть обработана и проанализирована. В большинстве случаев большие наборы данных обновляются в режиме реального времени по сравнению с ежедневными, еженедельными или ежемесячными обновлениями во многих традиционных хранилищах данных. Проекты по анализу больших данных принимают, сравнивают и анализируют поступившие данные, а затем дают ответ или результат на основе всеобъемлющего запроса [4].

Как сохраняются и обрабатываются большие данные

Необходимость обработки больших объемов данных предъявляет уникальные требования к базовой вычислительной инфраструктуре. Вычислительная мощность, необходимая для быстрой обработки больших объемов и типов данных, может разрушить один сервер или кластер серверов. Организации должны использовать эквивалентную вычислительную мощность для задач, имеющих большие данные, чтобы достичь необходимой скорости. Это может потребовать сотни или тысячи серверов, которые могут обработать в кластерной архитектуре и работать совместно [4].

Достижение такой скорости является проблемой и экономически выгодным подходом. Многие руководители предприятий стараются вложить средства в широкую инфраструктуру серверов и складов для поддержания рабочих нагрузок больших данных, особенно не работающих круглосуточно. В результате общедоступных облачных вычислений в настоящее время является основным инструментом размещения больших систем данных. В целом, поставщик облачных данных может сохранить базы данных и увеличить необходимое количество серверов для выполнения проекта анализа больших данных. Бизнес платит только за фактически

использованное время хранения и подсчета, и вы можете удалить экземпляры облака до тех пор, пока это необходимо [5].

Для дальнейшего повышения уровня обслуживания, общедоступные поставщики облачных служб позволяют работать с большими данными через службы управления, которые включают в себя наиболее распространенные вычислительные экземпляры Apache Hadoop, механизм обработки Apache Spark и соответствующие большие технологии данных. Amazon Web Services (AWS) Amazon Elastic MapReduce (EMR) - один из примеров больших служб данных, работающих в больших облаках; другие включают Microsoft Azure HDInsight и Google Cloud Dataproc. Большие данные в облачной среде могут храниться в файловой системе Hadoop (HDFS) или Amazon в дешевом облачном хранилище, например Amazon Simple Storage Service (S3).

Для организаций, которые хотят развертывать большую локальную систему данных, как правило, Hadoop и Spark дополнительно открыты оригинальные технологии Apache, в том числе еще один инструмент согласования ресурсов (YARN), Hadoop-геменеджер ресурсов и планировщик задач, включенных MapReducев Hadoop; рамка программирования MapReduce; Kafka, HBase, Drill или могут заручиться поддержки таких платформ, которые предлагают компании Cloudera, Hortonworks и MapR Technologies.

Однако Cloudera и Hortonworks договорились присоединиться к октября 2018 года, что может сократить количество доступных локальных платформ до двух [6].

Большие данные можно сравнить с небольшими данными, еще одним развивающимся термином, который часто используется для описания данных, его объем и формат можно легко использовать для анализаторов самообслуживания. Аксиома часто упоминается «большие данные предназначены для машин; небольшие данные предназначены для людей» [6].

Объём цифровых данных растет с ошеломляющей скоростью. Многое мусор, как этот документ, но теперь и этот мусор стало проще обрабатывать и получать статистический анализ. А статистика — это царица наук. Например, мы проводим тест новой видеокарты. В каждой дисциплине от 5 до 10 раз. И то, что потом выходит, либо простое среднее значение, либо среднее взвешенное среднее, в зависимости от тестов. И чем больше тестов мы проведем, тем точнее получим результат. В нашем случае мы получим небольшие данные, а вот если Nvidia или AMD будет получать от каждого пользователя отчет, о том, какая у него производительность, то компания будет владеть совершенно точными информациями относительно видеокарты и это как раз будут большие данные.

Большие данные в финансовой сфере, особенно в сфере финансовых услуг, например, используются во многих приложениях:

Контроль и Мониторинг персонала;

Предполагающие модели, которые могут быть использованы страховыми

андеррайтерами и кредитными специалистами для определения премий для принятия решения о кредитовании;

Разработка алгоритмов прогнозирования направления финансовых рынков; оценка неликвидных активов, таких как недвижимость.

Финансовые учреждения широко используют большие данные: начиная с повышения кибербезопасности до снижения оттока клиентов, развитие лояльности клиентов и многих других за счет инновационных и персонализированных предложений, которые делают современные банковские услуги высокоразвитыми[6].

1.1.2 Как сохраняются и обрабатываются большие данные

Необходимость обработки больших объемов данных предъявляет уникальные требования к базовой вычислительной инфраструктуре. Вычислительная мощность, необходимая для быстрой обработки больших объемов и типов данных, может разрушить один сервер или кластер серверов. Организации должны использовать адекватную вычислительную мощность для решения задач, имеющих большие данные для достижения необходимой скорости. Это может потребовать сотни или тысячи серверов, которые могут обработать в кластерном архиве и работать совместно [6].

Проблемой также является достижение такой скорости экономически эффективным способом. Многие руководители предприятий, особенно не работающих круглосуточно, стараются вложить средства в широкую инфраструктуру серверов и складов для поддержания рабочих нагрузок больших данных. В результате общедоступных облачных вычислений в настоящее время является основным инструментом размещения больших систем данных. Общий поставщик облаков данные могут хранить петабайды и увеличивать необходимое количество серверов для выполнения проекта анализа больших данных. Бизнес платит только за фактически использованное время хранения и подсчета, и вы можете удалить экземпляры облака до тех пор, пока это необходимо [6].

Для дальнейшего повышения уровня обслуживания, общедоступные поставщики облачных служб Apache Hadoop, Apache Spark позволяют работать с большими данными через службы управления, которые включают в себя наиболее распространенные вычислительные экземпляры Apache Hadoop, механизм обработки Apache Spark и соответствующие большие технологии данных.

Amazon Web Services, ElasticMapReduce (EMR) от Amazon Web Services (AWS) - один из примеров больших служб данных, работающих в облачных облаках; другие включают Microsoft Azure HDInsight и Google Cloud Dataproc. Большие данные в облачной среде Hadoop могут храниться в файловой системе Hadoop (HDFS) или в дешевом облачном хранилище, например Amazon Simple Storage Service (S3); База данных NoSQL является еще одним типом лучших вариантов в облаке для приложений [6].

Развертывание системы больших данных для организаций, местного,

которого, как правило, Hadoop и Spark , которые есть приложение с открытым исходным кодом для Apache технологии, в том числе ресурсов еще один инструмент согласования (YARN), Hadoop-менеджер ресурсов и планировщик заданий, включенных в реестр; MapReduce рамки программирования; Kafka, приложения и направление потоков данных между платформой обмена сообщениями; HBase база данных; Drill, HiveImpala , как и SQL-on-Hadoop с открытым исходным кодом, пользователи могут самостоятельно установить механизмы и есть механизмы варианты запросов или технологий Cloudera, Hortonworks и MapR Technologies, предоставляемых на коммерческой платформы большой данных, могут обратиться в поддержку они оказывают в облаке. Однако, Cloudera и Hortonworks договорились о присоединении в октябре 2018, что может сократить количество доступных локальных платформ до двух [6].

Большие данные можно сравнить с небольшими данными, еще одним развивающимся термином, который часто используется для описания данных, его объем и формат можно легко использовать для анализаторов самообслуживания. Аксиома часто упоминается

«большие данные предназначены для машин; небольшие данные предназначены для людей " [6].

Большие данные в финансовой сфере, особенно в сфере финансовых услуг, используются во многих приложениях, например:

- мониторинг и контроль персонала;
- прогнозирующие модели, в которых страховые андеррайтеры могут использовать кредитные специалисты для определения вознаграждения и принятия решения о предоставлении кредита;
- разработка алгоритмов прогнозирования направления финансовых рынков;
- оценка неликвидных активов, таких как недвижимость.

Финансовые учреждения широко используют большие данные: повышения кибербезопасности до снижения оттока клиентов, развитие лояльности клиентов и так далее за счет инновационных и персонализированных предложений, приводящих к высокоэффективному виду современных банковских услуг [7].

1.2 Технологии и методы передачи данных на финансовой бирже

Сериализация данных — это процесс преобразования структурированных данных в формат, который позволяет хранить данные в формате, позволяющем совместить их или восстановить первоначальную структуру [8, 10].

Это процесс преобразования в формат, в котором можно сохранить структуру данных или состояние объекта (например, файл или передается в буфере памяти или канале сетевого подключения), а затем может быть восстановлен в той или иной компьютерной среде.

В некоторых случаях вторая цель сериализации данных заключается в уменьшении количества данных, а затем снижении требований к дисковому пространству или пропускной способности. В 2019 году существует более 60 видов форматов сериализации данных. Наиболее популярными и распространенными видами являются: JSON, XML, ASN.1, Protocol Buffers (protobuf), и многие другие [8].

Сериализация нарушает прозрачность типа абстрактных данных, раскрывая потенциал отдельных компонентов реализации. Тривиальная реализация всех данных может нарушить инкапсуляцию.

Для того чтобы убедить конкурентов не создавать совместные продукты, издатели проприетарного программного обеспечения часто держат в секрете детали форматов сериализации своих программ. Некоторые специально вводят в заблуждение или зашифруют данные, зашифрованные. Однако совместимость приложений требует понимания форматов сериализации друг друга. Поэтому архитектура удаленного вызова таких методов, как CORBA, детально определяет свои форматы сериализации.

Многие учреждения, такие как архивы и библиотеки, стремятся в будущем проверять мусор базы данных, сохраняя свои резервные архивы, в частности, в сериализированном формате, удобном для их изучения [8, 9].

1.2.1 Форматы сериализации данных или сравнение форматов сериализации

В начале 1980-х годов технология Xerox Network Systems Courier повлияла на первый широко принятый стандарт. Sun Microsystems в 1987 опубликовал «представление внешних данных " (XDR) [8]. В конце 1990-х годов на стандартные протоколы сериализации началось стремление дать альтернативу: XML использовался для создания удобного текстового кодирования. Такое кодирование может быть полезно для постоянных объектов, которые человек может прочитать и понимать или передавать другим системам независимо от языка программирования. Он состоит из потери компактного кодирования на основе потока байт, но в этот момент большие емкости хранения и передачи сделали более важным количество файлов, чем в первые дни подсчета.

Двоичный XML эти текстовые редакторы не читались, но были компактными, чем обычный XML. В 2000 году Ajaxв веб-приложениях XML Ajax часто использовались для асинхронной передачи структурированных данных между клиентом и сервером [8].

JSON-это текстовый эквивалент XML, который широко используется для подключения клиента к серверу в веб-приложениях. JavaScriptНа основе синтаксис JSON JavaScript, но поддерживается на других языках программирования.

Протокол YAML содержит сильные, удобные для человека» и потенциально компактные функции для сериализации, такие как JSON. Эти функции включают понятие маркировки типов данных, поддержку

неерархических структур данных, возможность структурирования данных отступом и множество форм скалярных цитат [8].

Формат сериализации, понятный еще одному человеку, - это NeXTSTEP, GNUstep и macOS Cocoaсписок свойств, используемых в macOS Cocoa [8].

Для большого объема набора научных данных, таких как спутниковые данные и исходные данные климата, погоды или океана, например, первая версия HDF, netCDF и GRIB разработаны четкие двоичные стандарты сериализации [9].

1.2.2 Поддержка языка программирования

Некоторые объектно-ориентированные языки программирования непосредственно поддерживают сериализацию объектов (или архивирование объектов) или с помощью синтаксических элементов или для обеспечения стандартного интерфейса для этого. Некоторые из этих языков программирования являются Ruby , Smalltalk , Python, PHP , Objective-C , Delphi, Java и семьей языков [10] .NET.

CFML позволяет идентифицировать структуру данных WDDX с тегами < cfwddx> и с помощью функции JSON SerializeJSON().

Стандартная библиотека OCaml обеспечивает маршализацию через модуль Marshal (его документы) и функции Pervasive input_value output_value. Когда язык программирования OCaml подвергается статической проверке, модуль Marshal может нарушить гарантию типа использования, так как нет способа проверки, рекомендует ли объекты ожидаемого типа маршализованный поток? OCaml функции или функции структура данных (например, форма, в которой имеется метод), содержащая, трудно перенаправить, так как код, выполняемый в функциях, не может передаваться по различным программам. (У функции позиционирования кода маршализма есть флаг, но он может быть удален только в одной и той же программе). Стандартные функции маршаллинга могут обрабатывать циклические данные, которые можно сохранить общий доступ и настроить флаг [10].

Некоторые Perlмодули Perl обеспечивают сериализацию CPAN механизмов , в том числе Storable, JSON: XS и FreezeThaw. Storable Perl содержит функции сериализации и десериализации для файлов Perl. Кроме прямого сериализации файлов, storable скаляр включает в себя функцию возврата нереализованных копий упакованных данных freeze и десериализацию такого скаляра.

Это полезно для передачи сложной структуры данных с помощью сетевого сокета или хранения ее в базе данных. При использовании Storable существует сеть безопасных функций, которые всегда сохраняют свои данные в формате, читаемом на любом компьютере с меньшей скоростью. Эти возможности называются N - store, N - freeze и т. д. Для десериализации этих структур не существует функций "n" - структур , сериализованных с функциями "N" и их машинно-зависимыми альтернативами, десериализованных постоянных thaw и retrieve.

C и C++ не обеспечивает сериализацию как какой-либо высокоуровневой структуры, но оба языка поддерживают любой тип встроенных данных, а также запись простейшей старой структуры данных в виде бинарных данных. Таким образом, написание пользовательских функций сериализации обычно тривиально .

Кроме того, такие решения, как ODM, система ORM для C++, основаны на компиляторе, способны автоматически создавать код сериализации с небольшими изменениями или без изменений в объявлениях класса. Другие популярные сериалы фреймворки-Boost.Boost Framework, Serialization, s11n фреймворк, Cereal. Структура многофункционального центра Microsoft также представляет свою методологию сериализации как часть своей архитектуры Document-View [10].

Delphi предлагает встроенный механизм для сериализации полностью интегрированных компонентов с IDE (также называемые стационарными объектами). Содержание компонента сохраняется в файле DFM и загружается в полет.

Java обеспечивает автоматическую сериализацию, которая требует установления интерфейса java.io.Serializable с помощью реализации объекта java. Реализация интерфейса обозначает класс "можно сериализировать", а затем редактирует сериализацию в Java. В интерфейсе Serializable не определены методы сериализации, но сериализованный класс является определенным, можно определить методы, содержащие специальные имена и сигналы, при их обнаружении, вызываются как часть процесса сериализации и десериализации. Программный язык также позволяет разработчику тщательно определить процесс сериализации посредством реализации другого интерфейса, Externalizableинтерфейс Externalizable, который включает два специальных метода, которые используются для сохранения и восстановления состояния объекта. Объекты по умолчанию не серийны и имеют три основные причины, по которым необходимо реализовать интерфейс Serializable для доступа к механизму сериализации Java. Во-первых, не закрепляет полезную семантику в случае, если все объекты не реализуются .

Например, объект Thread зависит от текущего состояния JVM. Thread объект поддерживает полезную семантику без десериализованного контекста. Во-вторых, сериализованное состояние объекта является частью соглашения о гармонизации его классов. Поддержка совместимости версий сериальных классов требует дополнительных усилий и внимания. Поэтому создание сериального класса должно быть намеренным дизайнерским решением, а не условием по умолчанию. Наконец, сериализация дает возможность доступа к неспециализированным отдельным классам, которые не доступны по-разному. Классы, содержащие конфиденциальную информацию (например, пароль) не должны быть сериализированы или экстернализированы. Стандартный метод кодирования использует рекурсивный перевод на основе сериализированных полей в потоке байтов и дескрипторы класса объекта. Примитивы, а также не переменные, не статические объекты кодируются в

поток . Все объекты ссылающие на объекты не отмеченные как Transient должны сериализоваться, если какой-либо объект в полной колонке содержит какой-либо объект, то сериализация заканчивается ошибкой. Разработчик может повлиять на это поведение, переопределяет переходность или сериализацию для объекта, поэтому некоторые части контрольного столба пересекаются и не сериализируются [10].

Java не использует конструктор для сериализации объектов. JavaВ настоящее время на базе данных "Java" можно будет реализовывать объекты Java с помощью JDBC и хранить их в базе данных. При свинге компоненты реализуют интерфейс Serializable, которые не гарантируют выносливость между различными версиями виртуальной машины Java. Таким образом, компонент Swing или любой компонент может быть сериализирован в поток байтов, но гарантирует, что он преобразуется на другом компьютере .

JavaFX-платформа на основе Java для создания приложений с графическим интерфейсом содержимого. Может использоваться как для создания настольных приложений, непосредственно запускаемых из операционных систем, так и для интернет-приложений (RIA), работающих в браузерах, так и для приложений на мобильных устройствах. JavaFX ранее использовалось для замены библиотеки Swing.

Версии Java 11 не входит в Java SE и не разрабатывается компанией Oracle (Gluon поддерживается как отдельный модуль) [1]. Но Oracle в качестве части Java SE 8 вносит необходимые изменения до 2022 года.

ECMAScript 5.1 в соответствии с требованиями, разработанными в настоящее время в Казахстане внедряются новые технологии, основанные на современных технологиях JSON.parse() и добавили методы stringify (). Хотя JSON изначально основан на подмножестве JavaScript, JSON JavaScript имеет недопустимые пограничные условия. А именно, JSON Unicode U + 2028 LINE SEPARATOR и U + 2029 PARAGRAPHSEPARATOR он показывает строки на экранных строках в кавычках, а версии ECMAScript 2018 и выше не делают его.

.NET Framework.NET Framework имеет несколько сериалов, разработанных Microsoft. Также у третьих лиц есть множество сериалов. Список постоянно растет.

PHP первоначально осуществляется через функцию serialize () и unserialize (), которая была введена в сериализацию. Кроме ресурсов PHP (файловые указатели, сокеты и т. д.) получает сериализацию любого вида данных. Встроенная функция unserialize () часто опасна при использовании совершенно ненадежных данных. Для объектов есть два "волшебных метода", которые класс можно реализовать внутри класса sleep () и wakeup () - из serialize () и unserialize () соответственно, это может быть восстановлено и восстановлено объект. Например, во время сериализации может потребоваться восстановление связи при закрытии связи с базами данных и десериализации; эта функциональность обрабатывается этими двумя магическими методами. Они также позволяют объекту выбрать какие свойства сериализации. Для объектов с PHP 5.1 Serializable интерфейс,

объектно-ориентированный механизм есть сериализация [10].

В целом, не курсивные и неделимые объекты storeOn могут быть сохранены и получены в удобном для человека объекте с использованием протокола storeOn:/ readFrom:. storeOn : SmalltalkSmalltalk создает текст выражения при оценке и использовании методов; ReadfromReadfrom восстанавливает исходный объект. Эта схема является особой в значении, в котором используются процедурные характеристики объекта, а не сами данные. Поэтому он очень гибкий, что позволяет классам определить компактные представления. Однако в своем первоначальном виде он не обрабатывает циклическую структуру данных и не соблюдает соответствие общих ссылок (т. е. две ссылки на один объект восстанавливаются как ссылка на две равные, но не одинаковые копии). Для этого существуют различные портативные и непереносимые альтернативы. Некоторые из них являются уникальными для конкретной реализации библиотек Smalltalk или класса. В Squeak Smalltalk есть несколько способов стерилизации и хранения объектов. Самые простые и наиболее часто используемые storeOn :/ readFrom : и двоичные форматы хранения, на основе smartrefstream сериализаторов. Кроме того, связанные объекты могут храниться и извлекаться с помощью Image Segments, оба "структуры хранения двоичных объектов" поддерживают сериализацию и вывод из компактной двоичной формы [10].

Оба обрабатывают циклические, рекурсивные и выделяемые структуры, сохраняют, получают информацию о классе и метаклассах и включают механизмы для перехода объекта в "полет" (т. е. для преобразования экземпляров, написанных старой версией класса с другим макетом объекта). Интерфейсы API аналогичны (storeBinary / readBinary), но детали кодирования разные, что делает их несовместимыми в двух форматах. Однако, Smalltalk / X-код является открытым исходным кодом и Smalltalks может быть загружен другими Smalltalks. Объекты Smalltalk сериализации не являются частью особенности ANSI Smalltalk. В результате, код сериализации объекта зависит от реализации Smalltalk.

Полученные бинарные данные тоже разные. Например, объект, созданный в Squeak Smalltalk, не может быть восстановлен в Ambrai Smalltalk. Поэтому приложения, работающие в нескольких реализациях Smalltalk, опирающиеся на сериализацию объектов, не могут совместно использовать данные между этими различными реализациями. Эти приложения включают в себя объектную базу данных MinneStore и некоторые пакеты RPC. Решение этой проблемы — это несколько пакетов Smalltalks, которые используют формат на основе XML для Smalltalks сериализации [10].

Как правило, структура данных Lisp может быть сериализирована с функциями "read" и "print". Например, переменная foo со списком массивов будет распечатана (print foo). Также объект read может быть прочитан из потока под именем s (read s). Эти две части реализации Lisp называются принтером и читателем. Человек читает, что выходит quot; printquot;, он использует списки в скобках, например: (4 2.9 » x " y).

В большинстве типов Lisp, включая Common Lisp, принтер не может определить весь тип данных, так как он не может предложить все типы данных. Например, принтер Common Lisp не может печатать объекты CLOS. Вместо этого программист может записать метод для универсальной функции print-object, который появляется при печати объекта. Это как метод, который используется в Ruby. Код Lisp записан в синтаксис читателя, который называется синтаксис чтения. Многие языки используют личные и различные Парсеры для работы с кодом и данными, Lisp использует только один. Файл с кодом Lisp может считаться как структура данных, может быть преобразован с другой программой, а затем, например, может быть выполнен или записан в цикле read-eval-print. Все читатели и писатели не поддерживают циклические, рекурсивные или делящиеся структуры [10].

Haskell сериализация Read и Show предназначены для видов, которые являются членами класса. Каждый тип, являющийся членом класса типа Read, получает данные из дорожки съемных данных определяет функцию. Класс типов Show, в свою очередь, имеет функцию show, которая может быть образована дорожным изображением объекта. Не нужно четко определить тип простой рекламы программисту функцию, он является произведением Read или производным show, или оба компилятора могут генерировать функции в соответствии со многими обстоятельствами (но не для всех: например, типы функций не могут автоматически принимать типы Show или Read) [10].

Автоматически разработанный экземпляр для Show создает действительный исходный код, поэтому значение можно активировать и форматировать код, созданный show, например, в интерпретаторе Haskell. Для эффективной стерилизации существуют библиотеки Haskell, которые позволяют выполнять высокоскоростную сериализацию в двоичном формате .

Windows PowerShell осуществляет сериализацию с помощью встроенного командлета Export-CliXML. Export-сериализирует объекты CliXML .NET и сохраняет полученный файл XML в файле. Для повторного создания объектов Import-CliXML используется командлет Import-CliXML, который создает десиализированный объект в экспортированный файл с XML. Объекты, которые часто известны как "собственные мешки", не являются живыми объектами; это изображения, которые обладают свойствами, но не имеют методов. Используя двухмерную структуру данных Import-CSV и Export-CSV можно сериализовать в формате CSV [10].

Julia осуществляет сериализацию с помощью модулей serialize(), которые предназначены для работы в той же версии Julia и/или в виде одной системы. Пакет представляет более устойчивую альтернативу, используя документированный формат и общую библиотеку, в которой упакована для различных языков, а формат сериализации разработан по умолчанию, а помните, с максимальной производительностью для связи по сети.

Python - язык программирования общего назначения. Синтаксис ядра Python минимально. Кроме того, стандартная библиотека охватывает

большой объем полезных функций.

Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные особенности-динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки особенностей, поддержка многостоточных вычислений, высокая степень структуры данных. В свою очередь, поддерживается выделение программ на модули, которые могут быть объединены в пакеты.

Python поддерживает динамическое типирование, т. е. тип переменной определяется только при выполнении. Поэтому лучше говорить о связывании значения с некоторым именем "вместо" назначения переменного значения. В Python существуют виды, внедренные: булавка, строка, Unicode-строка, целое число свободной точности, число с плавающей запятой, комплексное число и т. д. Pythonвключены из коллекции Python: список, кортеж (неизменный список), Словарь, множество и т. д. Все значения являются объектами, в том числе функции, методы, модули, классы.

Добавление нового типа (class) происходит путем определения нового типа (например, написанного на языке C) в модуле ввода или расширения класса (класса). Система классов поддерживает наследование (единственное и множественное число) и метапрограммирование. Возможно наследование от многих встроенных видов и типов расширений.

Все объекты подразделяются на ссылочные и атомные. Атомная int, long (в 3-м варианте int любое число, потому что в 3-м варианте нет ограничений на измерение), complex и другие. При передаче атомных объектов их значение переносится, а для ссылок только на объект переносится показатель, таким образом, после передачи оба переменные используют одинаковое значение. Формы ссылок будут изменены и изменены. Например, строки и Кортежи будут неизменны, а списки, словари и многие другие формы будут изменены. В Python Кортеж действительно неизменный список. В большинстве случаев кортежы работают быстрее, чем списки, поэтому лучше использовать их, если вы не планируете изменить цепочку. Python -активно развивающийся язык программирования, новые варианты добавления/изменения языковых свойств выходят примерно раз в два с половиной года. Язык не подвергался официальной стандартизации, роль стандарта де-факто выполняет CPython, созданный под контролем языкового автора [10]. Python основан на обычном двоичном кодировании(SBE).

Sbedecoder - простой пакет python для анализа кодированных данных SBEpython. sbedecoder XML динамично создает парсер SBE в формате XML. Это достигается путем создания sbeschema () экземпляра и приглашения его метода parse () с именем файла:

- *from sbedecoder import SBESchema schema = SBESchema() schema.parse('path/to/schema.xml').*

Это было частью языков объектно-ориентированного программирования, непосредственно поддерживающих сериализацию объектов и данных.

Есть несколько специальных IDE для разработки Python. JetBrains

PyCharm-Python доступен на полнофункциональных платформах IDE, Windows, Mac OS X и Linux, в бесплатных (Community) и платных (Professional) версиях. PyCharm-интегрированная развивающая среда для языка программирования Python. Инструмент анализа кода, графический регулятор, юнитзапуск юнит-тестов и поддерживает веб-разработку Django. PyCharm разработан компанией JetBrains на основе IntelliJ IDEA [10].

PyCharm Windows, Mac работает с операционными системами Windows, Mac OS X и Linux.

Возможности:

- статистический анализ кода, синтаксис и подсветка ошибок.
- навигация по проекту и исходному коду: отображение файловой структуры проекта, быстрый переход между файлами, классами, методами и методами .
- рефакторинг: переименование, вывод метода, введение переменной, введение константы, подъем и спуск метода и т. д. б.
- инструменты веб-разработки использование Django фреймворка құралдары
- встроенный регулятор для Python
- юнитинструменты, встроенные для юнита-тестирования құралдар
- разработка с помощью Google App Engine
- Поддержка системы управления версиями: Mercurial, Gitобщий пользовательский интерфейс для Mercurial, Git, Subversion, Perforce и CVS [10].

1.3 Архитектура SBE и поиск схем сообщений

1.3.1 Сериализация данных и десериализация

Сериализация в контексте хранения данных в области компьютерных наук - это процесс преобразования структуры данных или состояния объекта в формат, который может быть сохранен (например, в файле или буфере памяти) или отправлен (например, посредством подключения к сети) и впоследствии восстановлен (возможно, в другой компьютерной среде). В результате, когда ряд битов пересчитан в соответствии с форматом сериализации, его можно использовать для создания семантического одинакового клона исходного объекта. Для многих сложных объектов, которые широко используют ссылки на Процесс, это не так просто. Сериализация объектно-ориентированных объектов не включает ни одного из ранее связанных методов. Этот процесс сериализации объекта называется маршалингом объекта или "marshaling". Обратная операция, получающая структуру данных из серии байтов, является десериализации (« unmarshalling») [11]. Используется:

- способ передачи данных по проводам (обмен сообщениями));

- способ хранения данных (в базе данных, на жестких дисках));
- например, метод вызова удаленных процедур, таких как SOAP;
- разработка программного обеспечения на основе таких компонентов, как метод распределения объектов, особенно COM, CORBA и т.д;
- метод определения изменений переменных данных.

Для того, чтобы некоторые из этих функций были полезными, необходимо быть независимыми от архитектуры. Например, для максимального использования распространения компьютера, работающего в другой аппаратной архитектуре, должна быть возможность надежного восстановления потоков сериализированных данных независимо от режима байтов. Это означает, что простая и быстрая процедура прямого копирования структуры памяти структуры данных не может работать надежно для всей архитектуры. Настройка байтов в формате, независимо от архитектуры структуры данных, установление памяти или структуры данных на различных языках программирования. означает исключение различных способов представления [11].

Поскольку кодирование данных по характеристике к любой схеме сериализации является последовательным, для получения части структуры сериализированных данных требуется от начала до конца прочтение и реконструкция всех объектов. В большинстве приложений это сетевое преимущество, так как он позволяет использовать обычные общедоступные интерфейсы ввода-вывода для сохранения и передачи состояния объекта. Высокая производительность в важных приложениях может иметь смысл приложить больше усилий для решения сложной нелинейной организации хранения.

Даже для того, чтобы сохранить простые объекты на одном компьютере, это слишком нежные показатели, так как они могут быть загружены в другое место в памяти отображаемых объектов. Для выполнения этого процесс сериализации включает в себя период, называемый распадом или распадом показателя, в котором прямые ссылки указателя превращаются в ссылки, основанные на именах или позициях. Процесс десериализации включает обратный шаг, который называется "swizzling" [11].

Ввиду того, что Сериализация и десериализация основаны на общем коде (например, функция Microsoft Foundation Classes Serialize), общий код может одновременно выполнять другие функции и таким образом, во-первых, выявлять различия между сериализированными объектами и объектами, которые являются их предыдущими копиями, и, во-вторых, обеспечить внедрение такого следующего определения. Не нужно создавать предыдущую копию, так как различия могут выглядеть во время перевозки. Эта техника называется *дифференциальным выполнением*. Это полезно при программировании пользовательских интерфейсов, содержание которых изменяется в течение времени-для выполнения этих задач можно создавать, удалять, редактировать или создавать графические объекты для обработки входящих событий без записи индивидуального кода [11].

1.3.2 Кластеризация данных

Класстеризация это процесс разделения обьектов по определенным признаком. Важно то что, когда мы занимаемся класстеризацией даныых, мы не знаем о том, к кому класстеру относятся данные. Класстеринг это группа похожих обьектов. У нас есть изначально не класстеризованные данные в виде точек. Мы проходимся по этим точкам и разбираем их уникальные параметры и основываясь на этих параметрах мы определяем их в ту или иную группу. Например, набор разных людей, врачи, клерки, бизнесмены, рабочие и тд. Мы знаем о них какую-то информацию. Допустим мы знаем их пол, возраст, район, где они живут, есть ли у них высшее образование или нет, какую музыку они любят и основываясь на этих данных мы можем запустить алгоритм кластеризации и разделить этих людей на группы, как в рисунке 1.0. Кластеризация это и есть процесс выявление определенных признаков о обьектах, определение их в группы и создание классов

Проблемы классификации требуют, чтобы данные были классифицированы по различным категориям. Используют помеченные данные

Проблемы кластеризации имеют наблюдения, разделенные на подмножества на разные подмножества, каждое из которых содержит аналогичную информацию. Эти подмножества называются кластерами и используют немаркированные данные

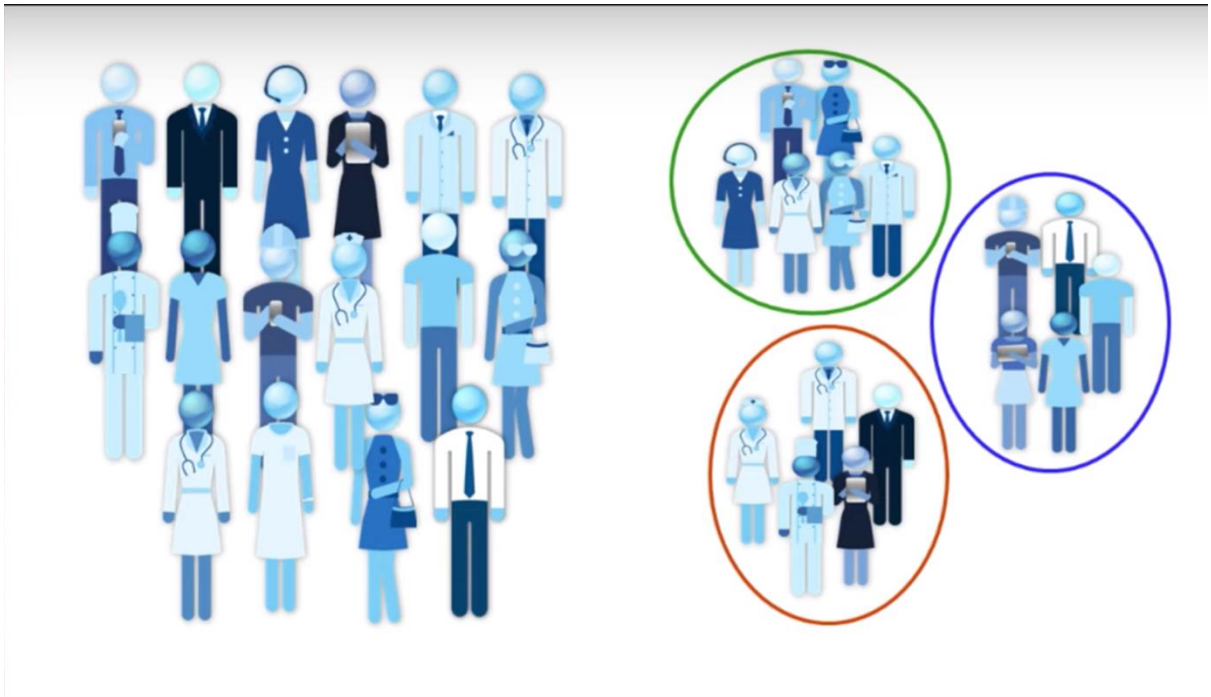


Рисунок 1.0 – Кластеризация на примере группы людей

Стоит задача, есть некоторые данные и есть алгоритм, это может быть мозг человека или алгоритм машинного обучения, которому нужно

определить на основе этих данных, может быть проект успешным или он провалиться. Таким образом мы подаем данные о проекте, и алгоритм машинного обучения классифицирует его к классу success или fail.

В случае с кластеризацией у нас есть три типа данных, это tower range – область покрытие, допустим те же LTE, 4G интернетом, вышка которая покрывает зону, есть local geography – данные о городе или где расположены обьекты и дальше у нас есть population – данные о плотности населения. И алгоритм кластеризации может помочь нам определить места в этих районах, куда лучше всего, оптимален и выгоднее всего поставить вышку, чтобы покрыть масштабную территорию.

Кластеризация и применения – retailing, marketing; insurance – страхование, banking – банковские операции; medicine – медицинская отрасль; biology – биологические эксперименты; publications – публикации.

Есть алгоритмы кластеризации.

Кластеризация на основе разделов - относительно эффективные (такие как K-средние, K-срединные, нечеткие C-средние)

Иерархическая кластеризация - продукты ветвяных (trees) кластеров (таких как агломерационные, дивизионные)

Кластеризация на основе плотности - создает кластеры произвольной формы (например, DBSCAN)

1.3.3 Fix SBE обмен финансовой информацией

eXchange протокол "финансовая информация в exchange (FIX)" является одним из наиболее часто используемых электронных протоколов связи [12].

Протокол FIX широко используется как при покупке, так и при продаже финансовых рынков. Среди его пользователей-взаимные фонды, инвестиционные банки, брокеры, фондовые биржи и ECN [12].

Большую часть электронных торговых протоколов составляют семьи протоколов FIX. В основном они используются электронными обменниками на финансовых биржах в Америке и Европе, в небольшой части азиатских стран.

Последние разработки и обработки протоколов протоколы FIX Simple Binary Encoding быстро входят протоколы отправки данных большого объема [12].

В настоящее время, в связи с тем, что в настоящее время в связи с тем, что в настоящее время в связи с увеличением количества абонентов, количество абонентов, проживающих в сельской местности, уменьшается на 10%. Binary Encoding (SBE) – высоко эффективное двоечное кодирование в сфере передачи данных. [13].

SBE FIX является частью группы протоколов, созданной рабочей группой высокого уровня в торговой ассоциации. SBE-протокол уровня представления OSI 6 для кодирования/декодирования сообщений в двоичном формате для поддержки приложений с низкой задержкой [13].

FIX Trading CommunityВерсия протокола электронной коммерции FIX,

поддерживающего высокопроизводительные транзакции и каналы данных в корпусе стандартов FIX Trading Communityжасақтаманы, официально запустила орган стандартов FIX Trading Community, чтобы помочь компаниям поддерживать высокоскоростное оборудование, программное обеспечение и сетевые соединения Trading Community[13].

Официальные источники утверждают, что новая версия оптимизирована для реализации с меньшей задержкой, включая кодирование и декодирование сообщений на десятки наносекундов. SBE протоколы очень прибыльные в компаниях таких как CME Group, Moscow Exchange, Thomson Reuters. Новая версия FIX Real Logic содержит версии с открытым исходным кодом, разработанные Real Logic для библиотек сообщений .

Протокол SBE обеспечивает различные характеристики, чем другие двоичные кодировки. Он оптимизирован для низкой латентности.

Версия 1.0 проект стандарта повысился от глобального технического комитета 9 февраля 2016 года до версии технической спецификации SBE 1.0. Это последняя особенность версии 1.0. Стандарты SBE Git доступны на сайте Git Hub и FIX Trading Community. Спецификация версии SBE 1.0 site была опубликована 27 июля 2018 года [13].

Версия 2.0 RC1 была одобрена глобальным техническим комитетом 16 августа 2018 года для 90-дневного общественного обзора. Возможности подключены благодаря популярному спросу. Поскольку он не совместим с версии 1.0, он был разработан в качестве основного варианта, поскольку он был незначительным изменением формата проволоки [13].

Используются размеры коротких сообщений, но обработка данных упрощает стоимость. Поддерживает FIX все семантики. Стандарт кодирования дополняет другие стандарты FIX для протокола сеанса и дополнительного уровня.

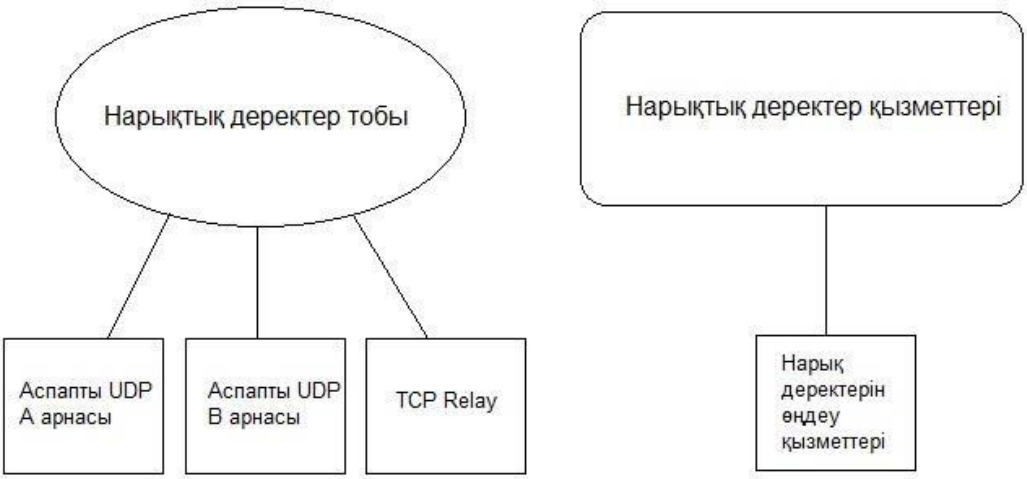


Рисунок 1.1 - Оптимизированная архитектура рыночных данных SBE

Рыночная группа данных. UDP A и B. Канал UDP-A и канал UDP-B используются для передачи инкрементных рыночных данных с использованием зашифрованных сообщений FIX SBE. Все типы сообщений FIX Feed A и UDP Feed отправляются через рыночные группы данных, которые используются в UDP Feed A и UDP Feed B. Это уменьшает вероятность потери сообщений, связанных с бинарным UDP. Каждое сообщение SBE передается по двум каналам (рисунок 1.1) [13].

Восстановление TCP Replay. Клиентские системы могут восстановить определенные сообщения, отправленные с использованием порядкового номера и компонента прогноза TCP. Исторический компонент воспроизведения TCP позволяет системам запросить повторение сообщений, опубликованных в канале данных инкрементного рынка UDP. Отображает сообщения для осуществления запросов. SBE Market Data Request (тег 35-MsgType = V) [13].

Этот тип запроса отправляется с помощью новых подключений TCP, установленных клиентскими системами. Ответы будут отправлены в CME Group по тому же сообщению и после завершения пересылки соединение CME Group закрывается. Все ответы кодируются протоколом SBE (включая отклонение).

При запросе сообщений через TCP Historical Replay используются следующие ограничения:

В запросе рыночных данных (35=V) может быть запрошено не более 2000 сообщений.

Запросы текущего дня могут быть запрошены и отправлены повторно [13].

1.3.4 Пользователи финансовой информации

Финансовая информация EXchange (FIX SBE) является нейтральной для международного обмена операциями по обмену ценными бумагами в реальном времени в отношении поставщиков электронного коммуникационного протокола, касающегося поставщиков, что полезно для фондов, инвестиционных менеджеров и фирм.

Системы FIX SBE предоставляют достоверную и своевременную финансовую информацию о сделках с ценными бумагами через валютно-обменные фонды и между ними. Его использование позволяет пользователям принимать своевременные и точные решения.

FIX SBE стал стандартом де-факто обмена сообщениями для торговли, торговли и послепродажного обмена, а также для нормативного отчета США. Он совместим со всеми распространенными сетевыми технологиями [14].

FIX Protocol, Ltd. Корпорация владеет и поддерживает систему FIX. Компания была полностью создана для достижения этой цели и обеспечения сохранности системы в общественном достоянии [14].

Соединение FIX включает текстовые сообщения и разделение электронной почты, ценных бумаг, новости, заявки и изменения, рекламу и

отчеты. В основном используется для взаимодействия между предприятиями, который предназначен для улучшения деловых сообщений и потоков транзакций. FIX осуществляет работу по достижению этой цели, уменьшению лишних функций и сокращению времени на телефонную связь, письменные сообщения, транзакции и документы.

В 1992 году между Salomon Brothers и Fidelity Investments внедренный для продажи акций протокол FIX станет стандартным для обмена опционами и фьючерсами. Он осуществлялся для обеспечения более эффективных и подотчетных транзакций и ведения записей, которые в основном телефон заменили систему, обработанную по телефону. В старой системе часто признаки интереса потерялись в режиме "ожидание" или были отправлены в неверный трейдер. FIX стал стандартным электронным протоколом для торговли и осуществления торговли [13]. Финансовая телекоммуникация среди мировых банков (SWIFT)

принимая во внимание, что бэк-офис является стандартом обмена сообщениями, FIX является стандартом обмена сообщениями фронт-офиса [14].

Отвечает: Кулик Мария Викторовна нет соц. налог не отчисляется, так как сотрудник в отпуске без содержания начислений у него нет, соответственно и отчислений тоже. Пользователи включают взаимные фонды, инвестиционные банки, брокеры, фондовые биржи и другие сети электронной связи (ECNs) .

В основном она используется для операций с акциями, но может обрабатывать операции с облигациями, иностранной валютой и деривативами [15].

FIX Trading CommunityСовместные усилия фирм-членов FIX Trading Community ™ поддерживает стандарт обмена сообщениями FIX и продолжит дальнейшее развитие данного протокола.

Члены ассоциации FIX включают в себя несколько ведущих финансовых учреждений по всему миру. Работа этих фирм-членов гарантирует, что стандарт развивается для удовлетворения новых и возникающих торговых условий. Их действия способствуют внедрению FIX во всем мире. Сам протокол FIX является непатентованным, бесплатным и открытым стандартом [15].

FIX-стремится быть информированным о стабильном, переменном значении, а также об изменениях в отрасли и технологиях. По состоянию на начало 2018 года участники обсудят текущие проблемы и вопросы, включающие кибербезопасность, цифровую валюту и блокировку, прозрачность исполнения и повышение производительности.

Fix (SBE) простая двоичная кодировка предназначена для высокопроизводительных торговых систем, но также подходит для других высокопроизводительных приложений. Он оптимизирован для кодирования и декодирования с низкой задержкой [15].

Эта особенность кодирования описывает проводный протокол для сообщений (уровень видимости). Таким образом, он обеспечивает стандарт

для взаимодействия между взаимодействующими сторонами. Пользователи могут использовать этот стандарт таким образом, чтобы они лучше соответствовали своим потребностям. Стандарт кодирования является дополнением к другим стандартам FIX для поведения протокола сеанса и уровня программы. SBE FIX предназначен для представления всей семантики.

Бинарная система типов. FIX поддерживает все виды документированных полей для поддержки традиционной семантики. Однако вместо печатаемых символьных изображений кодирования *fix система типов fix tag = value* связывается с двоичными типами данных и при необходимости определяет производные [15].

Система бинарных видов была усовершенствована следующим образом:

Предлагает инструменты для отображения точности десятичных чисел и знаков времени, а также диапазонов разрешенных чисел. Этот метод отделяет символические массивы заданной длины от строк переменной длины.

Большинство вариантов обеспечивают согласованную систему цепей и полей, которые существуют.

Принципы дизайна:

Проект SBE стремится к непосредственному доступу к данным без сложных преобразований или условной логики. Это достигается посредством:

Использование двоичных типов и простых видов данных, полученных из собственных двоичных файлов, таких как цены и временные знаки.

Необходимо поддерживать доступ к прямым данным, которые должны быть последовательно обработаны, и поддерживать необходимость управления множеством элементов в длине переменные поддерживающая, фиксированная позиция и поля поддерживают преимущество фиксированной длины [15].

Схема сообщения. Этот стандарт описывает, как кодируются поля и структуру общего сообщения. Содержание вида сообщения определяется схемой сообщения. Схема сообщения сообщает, какие поля сообщений принадлежат и их расположение в сообщении. Кроме того, метаданные описывают диапазон допустимых значений и информацию, которая не передается в провода, например постоянные значения.

Схемы сообщений могут основываться на стандартных особенностях сообщений FIX или могут быть настроены в соответствии с соглашением между контрагентами.

Не нужно одновременно обновлять всех издателей и пользователей сообщений. В рамках определенных ограничений схемы сообщений и форматы проволоки могут быть расширены в контролируемом виде. Если для обработки бизнеса не требуется интерпретировать подключенные поля или сообщения, потребители, использующие старую версию схемы, должны быть совместимы.

Стандарт простого бинарного кодирования разработан некоммерческим отраслевым органом по стандартизации, расположенным в мировом торговом центреTrading Community, высокопродуктивной рабочей группой FIX Trading

Community [15].

1.3.5 Криптовалютные данные и его значимость

Понятие "криптовалюта" относится к количественным видам валют. Его делают и дают с помощью криптографических методов, в основном, на основе блокчейнтехнологии blockchain. "Монеты" выводятся первоначально за счет проведения математических расчетов в электронном виде. Простые слова, криптовалюта-это искусственная платежная система, которая равняется реальным деньгам с официальным курсом [16].

Криптовалюта - это еще одна технология, способная изменить мир. Это понятие появилось несколько лет назад. Однако технология была очень полезной и востребована в различных отраслях. От ежедневных платежей до защиты национальных интересов.

История создания криптовалюты Bitcoin началась с создания биткоина. Биткоин - самый первый этап криптовалюты. Он является самым популярным и дорогим на сегодняшний день [16].

Рисунок 1.2 - Биткоин

Все началось в самый разгар финансового кризиса 2008 года. В том же году группа энтузиастов решила создать универсальное платежное решение, которое не будет зависеть от политики. 31 октября 2008 года на одном из сайтов для программистов появилась статья [17]. Она называлась "Биткойн : электронные наличные в пиринговой системе". 9 января 2009 первая версия биткоина-кошелька была выпущена и появилась первая версия биткоина. На самом деле-это первые криптовалюты.

В течение нескольких месяцев была проведена определенная проработка этой системы. Появилась поддержка многих различных параметров и

популярной linuxоперационной системы linux. В тот момент биткоин известен лишь группе его разработчиков и тестеров. Поэтому разработчики системы решили привлечь внимание общественности. В ноябре 2009 года был открыт форум по расчетам с биткоинами. Популярность биткоинов стала немного расти. Кроме того, в обсуждении появились новые идеи по улучшению первой криптовалюты.

В 2010 году разработчики обнаружили и уничтожили несколько уязвимостей криптовалюты. Увеличена скорость передачи биткоинов. После этого количество биткойн-кошельков стало расти. В мае 2010 года состоялся первый обмен биткойнов реальным товаром [17].

Затем американский Ласло Ханеч заменил 10 000 биткойн на две пиццы. На самом деле, это было первое использование биткоинов в качестве денег. С 2012 года возглавляет американская Bitcoin компания Bitcoin Foundation, обеспечивающая развитие Bitcoin. Главный разработчик этой компании- Гэвин Андресен . Именно сейчас его можно считать руководителем проекта. Известно, что на сегодняшний день существует более 250 различных криптовалют. https://ttrcoin.com/coins источник: Состояние криптовалютной биржи в мае 2019 года (рисунок 1.3) [18].

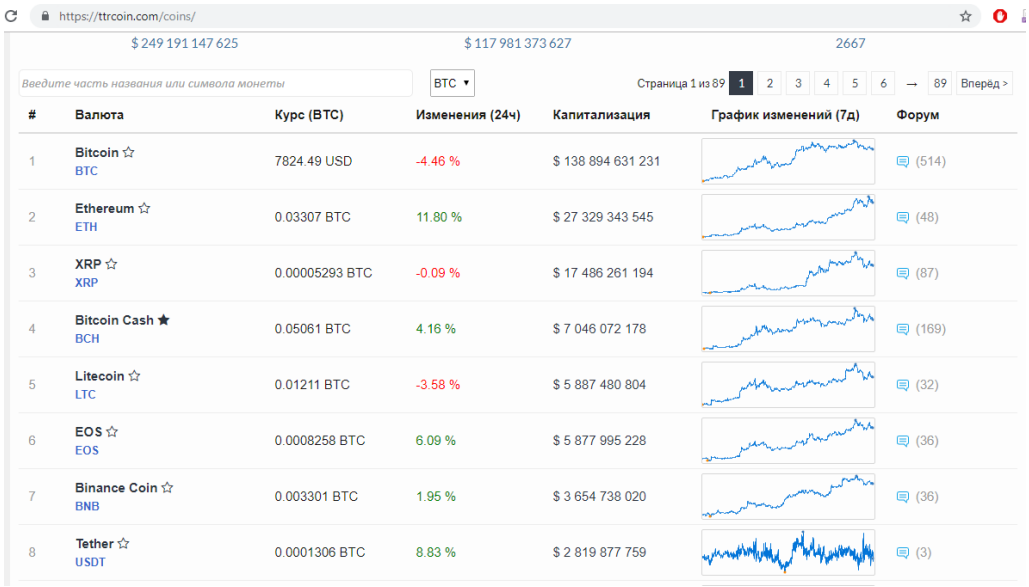


Рисунок 1.3 - Криптовалютаныңкурсы криптовалюты

Новая криптовалюта выпускается в цифровом виде. Любой человек может получить криптовалюту. Процесс производства новой криптовалюты называется майнинг. Для занятия майнингом используется вычислительная мощность компьютера. Слово "Майнинг" пришло к нам из английского языка. Переводится как "добыча полезных ископаемых".

Термин "криптовалюта" впервые Forbesначал обсуждаться в 2011 году с публикации журнала Forbes. С тех пор название прочно вошло в доход и применяется к коэндам (дословным монетам), не имеющим выражений в виде бумажных банкнот или монет из металла. Такой вид денег существует только

в цифровом пространстве.[17]

В отличие от других электронных платежных систем, криптовалюты первоначально появляются без реального платежа. Чтобы стать владельцем определенной суммы коинов, достаточно подключиться к сервису их создания, стать участником Единой майнинговой сети и ждать своего "успеха". Основное отличие этой криптовалюты от реальных денег-последнее выпускается строго по решению Центрального Банка РФ. https://www.rbc.ru / диаграмма курса продажи биткоина за последние 5 лет (рис.1.4) [18] с официального сайта.

Рисунок 1.4 - Биткоиннынрост курса биткоина за последние 5 лет

За последние десять лет криптовалют играет важную роль в области информационных технологий. В связи с этим наблюдается быстрое увеличение количества переданных и принятых данных и возникает необходимость увеличения скорости передачи данных. В соответствии с этим, следует отметить практическую значимость исследования, что разработанная программа обеспечивает возможность снижения задержек в получении информации и оперативной передачи большого объема данных. Распространение информационных технологий во многих регионах нашей страны очень высока. Фондовая биржа криптовалют является одной из этих отраслей. Фондовый рынок сегодня является одной из высокотехнологичных отраслей. А на основе этого прогресса создается несколько торговых площадок, брокерских систем, которые борются с высокой нагрузкой высокоскоростных каналов связи. Кроме того, для передачи большого объема информации спрос на приложения, обеспечивающие большую компрессию

данных, имеет актуальность не только в финансовой системе, но и в маркетинге, бизнесе, телекоммуникациях, торговле, логистике и правительстве. Согласно статистике, объем сохраненных и полученных данных ежегодно растет, при этом очень важен вопрос передачи быстрых данных и играет большую роль в XXI веке. Научная новизна работы криптовалюта- разработка метода снижения оперативной задержки финансовых криптовалютных данных большого объема, что позволяет хранить декодированные и десериализованные данные в базе данных и представлять в виде японских ламп [17].

1.3.6 Принципы работы биржевых графиков

Биржевые графики являются важным инструментом рыночного анализа, так как они представляют информацию о цене в графическом, визуальном варианте, что является более простым для общего понимания ситуации на рынке, чем текстовая или цифровая информация. Биржевые графики позволяют увидеть массовое поведение группы, а также оценить расположение сил между продавцами и покупателями, что в конечном итоге позволяет понять потенциал доходности сделки [19].

Виды биржевых графиков. Существует три основных вида графической стоимости-линия, бар и японская лампа. Каждый из этих средств предоставляет информацию о цене открытия соответствующего периода, о цене закрытия, о минимальных и максимальных ценах (кроме сетевого графика, создаваемого только на основе цены закрытия). Графики, используемые трейдерами, свидетельствуют об изменении цены за определенную часть времени, например, за 5, 15, 60 минут, за день, неделю или месяц.

Историческая биржевая графика не является другой вещью, как история борьбы между конкурентами, доминирующими на рынке. В результате такой борьбы заключается большое количество сделок, каждый из которых обязательно указывается в таблице - один ИИК соответствует такой сделке. Когда цена растет, это означает, что кто-то понесет убытки, а кто-то заработает.

Такие модели могут быть обратными, т. е. после их появления цена может изменить свое направление на противоположные или продолжающие фигуры, т. е. в результате их формирования цена будет и дальше двигаться в том же направлении (примеры ценовых формаций-голова и плечо, двойное дно, три конца, флаг и флаг, треугольник и другие).

Биржевые таблицы являются важным инструментом анализа текущей рыночной ситуации и прогноза дальнейшего поведения цены. Они позволяют определить баланс сил между быками и медведями и сделать выбор в пользу кого-то. Кроме того, биржевая таблица может помочь оценить потенциал роста (или падения) по инструменту, анализируемому для принятия решения о вступлении в сделку [19].

Исследование технического анализа начинается с них. модели могут быть обратными, то есть. после их появления цена может изменить свое направление на противоположные или продолжающие фигуры, т. е. в результате их формирования цена движется и дальше в этом направлении (примеры ценовых формаций – голова и плечо, двойное дно, три кончика, флаг и флажок, треугольник и т. д.). В биржевой торговле в настоящее время используются три основных вида таблиц: линейные, барные и японские лампы.

Линейная таблица. Самый простой и понятный для представления биржевых котировок. Именно такие графики все рисовали в школе, институте. В таблице проставляются точки, равные цене закрытия этапов. И эти знаки соединяются с линиями. Линейная таблица человек удобнее принимать, так как в нем нет дополнительной информации. Все очень просто-линия вверх-тренд роста, падение-падение. Однако, для существенного анализа, там не хватает более дополнительной информации, поэтому многие трейдеры используют другие виды графиков.

Бары. Графическое представление цены в таблице с использованием баров более информативно. В отличие от сетевых баров, бары указывают дополнительную цену открытия, максимальную и минимальную цену за данный период (рисунок 1.5).

Вертикальная линия с поперечными отрезками справа и слева. В зависимости от того, какой был выбран интервал отображения графиков, такие данные указывают один путь. Если выбран дневной интервал, то один бар — один день, недельный-1 неделя и т. д. [19].

Рисунок 1.5 - Таблица в виде баров

Длина баров соответствует диапазону цен. Верхние и нижние точки - это самый высокий и самый низкий уровень котировок. Правая линия-цена открытия, левая-закрытие. Если левая линия ниже правого, то цена закрытия выше цены открытия и мы наблюдаем рост. Наоборот, когда левая линия справа выше, получаем цену закрытия, цена открытия и падение котировок на

рынке .

Японские свечи. Японские свечи, такие как бары, в частности, цена открытия и закрытия, несет совершенно аналогичную информацию, такую как максимальные и минимальные котировки за выбранный период. Разница между барами и лампами только в графическом представлении (1.6).

Если бары указаны в виде горизонтальной линии, то часть светильников между ценой открытия и закрытия будет утолщена. Его называют световым телом. А от тела свечи до максимума-тень свечи.

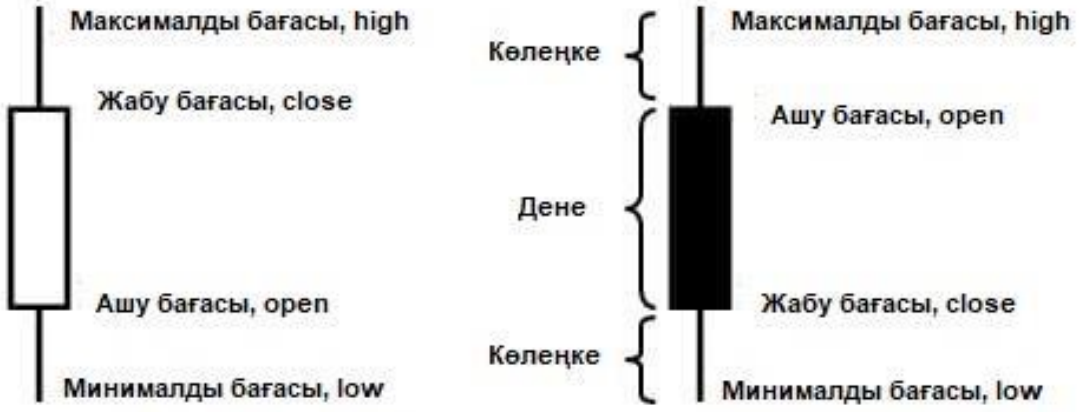


Рисунок 1.6 - Японские свечи

Рисунок 1.7 - Биткоин линейная и барная таблица курса биткоина кроме того, в зависимости от роста или падения японской свечи тело окрашивается в разные цвета

В классическом варианте белый цвет-рост, черный-падение. Другой распространенный вариант: красный говорит и Quotes уменьшает котировки сочетание зеленого и красного цветов, зеленый о росте. Многие трейдеры используют комбинации цветов, но это привычки.

Цветная окраска улучшает восприятие графика. На рисунке 1.7 https://www.ino.com / графики линейных и японских ламп биткоина с 12 по 14 мая 2019 года [19].

1.4 Актуальность и новизна исследования

Сегодня, в мире современных инноваций Информационные технологии играют важную роль во всех сферах деятельности. Объем данных передаваемой и получаемой информации показывает быстрый рост и возникает проблема необходимости увеличения скорости передачи данных. В соответствии с этим практическая значимость в работе позволяет уменьшить задержки в получении разработанной дополнительной информации и ускорить большой объем данных .

Распространение информационных технологий во многих районах страны очень высока. Фондовая биржа является одной из этих отраслей. Как известно, фондовый рынок в настоящее время является высокотехнологичной отраслью. На основе этих прогресса создается несколько торговых платформ, брокерских систем, которые выполняют большую нагрузку на высокоскоростные каналы связи. Следовательно, необходимость приложений, обеспечивающих большую компрессию данных для передачи значительного объема информации, очень высока не только в сфере финансовой системы, но и в маркетинге, бизнесе, телекоммуникациях, торговле, логистике и государственных управлениях.

Согласно статистике, весь объем хранимых и получаемых данных ежегодно растет, и проблема оперативной трансмиссии данных очень важна и играет важную роль в XXI веке. Научная новизна работы в разработке метода для оперативной передачи как минимум большого объема сниженных финансовых материалов, позволяющего хранить декодированные и десерированные данные в базу данных и представлять их в виде биржевых японских светильников.

2 ПРОСТОЙ ПРОТОКОЛ ДВОИЧНОГО КОДИРОВАНИЯ

2.1 История процессора и памяти Revolution

Простое двоичное кодирование (SBE) - это очень быстрая библиотека сериализации, используемая в финансовой индустрии [20].

Целью сериализации является кодирование и декодирование сообщений, и существует множество доступных опций, начиная с протокола XML, включая JSON, Protobufer, Thrift, Avro и так далее. б. применяется.

XML, JSON - это кодирование и декодирование на основе текста, что часто хорошо, но когда важна задержка, кодирование и декодирование на основе текста будет более узким.

Protobuffer, Thrift, Avro являются бинарными опциями и широко используются [20].

Простой Binary Кодирование двоичное и строится на основе механического сочувствия к использованию основного оборудования (кэш - памяти процессора, предварительного выбор, модель доступа, конвейерной инструкция и т.д.) [20].

В области современных технологий используются 8-битные, 16-битные, 32-битные, 64-битные процессоры. Сегодня купить сервер емкостью 512 ГБ очень просто из-за низкой стоимости памяти [20].

Большая часть системы опирается на оптимизацию во время выполнения, но SBE использует полностью верный подход, и первый уровень оптимизации выполняется компилятором.

Рисунок 2.1 - Простая двоичная кодировка сериализации

Как показано на рисунке 2.1, «схема» здесь предназначена для определения макета XML- файла и типа данных сообщения. «Компилятор» принимает схему в качестве входных данных и создает промежуточное представление (промежуточное представление или IR) - это структура данных или код, используемый компилятором или виртуальной машиной для представления исходного кода. У этого слоя есть много интересных применений, таких как последний / стабильный оптимизированный код. Далее сообщение - сообщение через покрытие буфера является метод стека tabıladı.Tolıq для оптимизации различных уровней .

Отсутствие ненужного кода очень важно для системы с низкой задержкой, если об этом не позаботятся, программа не сможет правильно использовать кэши ЦП и может переключиться на задержку GC.

SBE построен на основе модели flyweight , все дело в том, чтобы повторно использовать объект, чтобы уменьшить нагрузку на память JVM.

Он имеет понятие буфера и может использоваться повторно, кодер и декодер могут принимать буфер в качестве входных данных и работать с ним. Кодер и декодер не разделены или очень малы (т.е. в случае String).

Он предлагает использовать буфер прямого / выключенного режима для полного удаления изображения GC , которое можно разделить на уровне потока и использовать для кодирования и кодирования сообщений.

Фрагмент кода для использования буфера:

```
final ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4096);

final UnsafeBuffer directBuffer = new UnsafeBuffer(byteBuffer);


tradeEncoder        .tradeId(1)
        .customerId(999)
        .qty(100)
        .symbol("GOOG")
        .tradeType(TradeType.Buy);
```

2.1.1 Предварительная выборка кеш

В CPU установлено аппаратное устройство предварительной выборки . Предварительный выбор кэша - это метод, используемый компьютерными процессорами для повышения производительности за счет указания или быстрого извлечения данных из исходного хранилища для замедления локальной памяти .

Доступ к данным из быстрого кэша ЦП в более быстром порядке, чем в основной памяти.

Если алгоритм является потоковым, предварительный выбор работает очень хорошо , а используемые данные являются непрерывными в виде массива. Доступ к массиву очень быстрый, потому что он последовательный и предсказуемый.

Массив SBE используется в качестве основного склада, и поля упакованы в него.

Данные перемещаются небольшими партиями пути кеша, который обычно составляет 8 байт, поэтому, если приложение запрашивает 1 байт, оно получает 8 байт данных. Поскольку данные упакованы в массив, доступ к одностраничному содержимому массива предварительного выбора ускоряет доступ и обработку.

[20] .

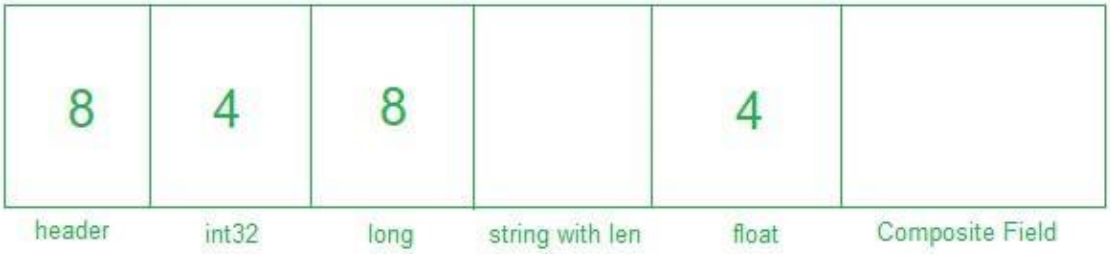


Рисунок 2.2 - Размеры типов данных в байтах

2.1.2 Потоковый доступ

SBE поддерживает все простые типы, а также позволяет определять пользовательские типы переменного размера, что позволяет иметь потоковые и последовательные коды и декодер . Он имеет хорошее преимущество чтения данных из пути кеша, и декодер должен знать очень мало метаданных о сообщении (то есть, E shift и измерение) [21].

Это особенно верно, если переменные типы данных закодированы,а процедура компромиссного чтения должна основыва ться на процедуре агрегирования .

Например, он используется под заказ, чтобы написать

```
tradeEncoder        .tradeId(1)
        .customerId(999)
        .tradeType(TradeType.Buy)
        .qty(100)
        .symbol("GOOG")
        .exchange("NYSE");
```

Для строковых атрибутов (символ и обмен) порядок чтения должен быть первым символом, затем, если приложение изменяет порядок, оно будет читать неправильное поле, а другая длительность чтения должна быть только один раз для атрибута переменной , поскольку это шаблон потока доступа [21].

2.1.3 Безопасный API-интерфейс

Проверка, связанная с массивом, может включать накладные расходы, но SBE использует рискованный API, и это не проверяет дополнительные накладные расходы.

Когда компилятор генерирует код, он рассчитывает материал заранее и использует константы. Примером этого является, когда выключено множество поля не учитывается в сгенерированного кода [21].

Каждое ядро имеет несколько портов для параллельной работы и несколько инструкций. Компилятор SBE генерирует бесплатный код из этих дорогих инструкций и имеет базовую математику с указателями.

Бесплатный код из дорогого руководства очень быстрый и может использовать все порты ядра [21].

Шаблон кода для сериализации Java

```
public void writeFloat(float v) throws IOException {
    if (pos + 4 <= MAX_BLOCK_SIZE) {
        Bits.putFloat(buf, pos, v);        pos += 4;    } else {
        dout.writeFloat(v);    }
}

public void writeLong(long v) throws IOException {
    if (pos + 8 <= MAX_BLOCK_SIZE) {
        Bits.putLong(buf, pos, v);        pos += 8;    } else {
        dout.writeLong(v);    }
}

public void writeDouble(double v) throws IOException {
    if (pos + 8 <= MAX_BLOCK_SIZE) {
        Bits.putDouble(buf, pos, v);        pos += 8;    } else {
        dout.writeDouble(v);    }
}
```

Code Шаблон кода VEDA для сериализации Java

```
public TradeEncoder customerId(final long value)
{
    buffer.putLong(offset + 8, value, java.nio.ByteOrder.LITTLE_ENDIAN);    return this;}

public TradeEncoder tradeId(final long value)
{
    buffer.putLong(offset + 0, value, java.nio.ByteOrder.LITTLE_ENDIAN);    return this;}
```

Размеры сообщений разных классов: маршал. Сериализуемый Маршал Размер класса - 267; маршал. ExternalizableMarshal размер класса - 75; Размер класса marshal.SBEMarshall составляет 49. SBE является самым компактным и очень быстрым, его авторы говорят, что он примерно в 20-50 раз быстрее, чем протокол Google Protobuffer.

2.2 Анализ и демонстрационный анализ методов и протоколов передачи данных

Основной особенностью исследования является использование протокола SBE для быстрого и компактного кодирования больших объемов данных. В настоящее время финансовая система работает, отправляя и получая большое количество сообщений в различных форматах. Каналы рыночных данных от финансовых бирж могут принимать десятки или сотни тысяч сообщений в секунду, и объем передаваемых данных постоянно увеличивается. В наше время FIX играет важную роль для получения рыночных данных, использующих кодировку символов ASCII, таких как тег FAST . К ним относятся хорошо известные протоколы XML и JSON в этих областях.

Современные электронные биржи не имеют возможности совершать транзакции быстрее, так как, согласно веб-сайту FIX, 75% фирм опрошены на биржах покупателями, а 80% - продавцами, обмен финансовой информацией для электронной торговли, начатый в 1992 году использует протокол. «FIX» ProtoBuf и FAST современные двоичные протоколы, такие как сериализация и дессериализация, скорость намного ниже.

Ранее FIX был предназначен для автоматизации коммуникаций и отправки финансовых данных на фондовый рынок. Однако протокол FIX был не самым надежным средством передачи увеличенных объемов финансовых данных, поэтому для его дальнейшего развития был разработан новый стандарт - FAST (FIX Adapted for STreaming).

Есть и другие решения таких проблем. Самым популярным из них является Protocol Buffer. Протокол Буферы является языком описания данных, предоставленных Google авторов в качестве альтернативы к XML. Разработчики убеждены, что Protocol Buffer гораздо удобнее, функциональнее и быстрее, чем XML.

Шломи Марко, представитель Goldfish and Investment Management, сказал, что он работает в фонде, который ведет высокочастотную торговлю. Они согласуются с точки зрения производительности - все рассчитывается до задержки в миллисекундах. При отправке сообщения они используют буфер протокола Google, когда основной программой на сервере является Java, а клиентской частью часто является C #. Точно так же Protobuf использовался в работе.

Тем не менее, в последние годы, чтобы создать разработчикам FIX ссылку - Simple Binary Encoding перейти на новый FIX- используя стандарт разработан. По словам Мартина Томсона, конечным результатом использования принципов разработки SBE являются кодеки с очень низкой емкостью и в 16-25 раз большей мощностью, чем буфер протокола Google (GPB), который имеет проблему предсказуемой задержки. Это наблюдалось в микротестировании и применении приложений.

По сравнению с Protobuf простое сообщение с рыночными данными может быть закодировано или декодировано за ~ 25 нс , в то время как Protobuf потребуется 1000 нс для расшифровки этого сообщения теми же инструментами и оборудованием . XML и FIX все медленнее и медленнее. В

настоящее время наиболее распространенным форматом передачи данных для информационных программ является JSON.

JavaScript Object Notation (JSON) - это формат обмена данными, который легко обрабатывается и генерируется приложениями, а также прост в использовании для пользователей. Он основан на языке JavaScript. JSON - C, C ++, C #, Java, JavaScript, Perl, Python и многие другие, выполняемые в соответствии с используемыми языками программирования, не зависят от языка, например текстовый формат. Формат JSON также подходит для сериализации небольших структур данных в задачах обмена данными между браузером и сервером и между самими серверами JSON. В связи с этим данные SBE и JSON в данной статье.

Сравнительный анализ и тестирование проводились между различными методами передачи.

2.2.1 Процедура проектирования

Чтобы достичь самого низкого уровня рассеивания в меньшей степени, остановка, основы проектирования и участие в исследованиях, проектировании, комплексной установке имеет первостепенное значение. Если необходимо разработать основу для компромиссного решения, он обязан оказать помощь в принятии решения по отбору конкурирующих кандидатов. Результаты использования этих системных ресурсов можно отнести к исследованиям. Существует шесть основных принципов проектирования, которые имеют большое значение в исследовании SBE. Первый называется *бесплатное копирование*. Сети и системы хранения работают напрямую с зашифрованными и расшифрованными буферами сообщений.

Правило независимого копирования заключается в том, что промежуточные буферы нельзя использовать для кодирования и декодирования. При использовании буфера затраты увеличиваются за счет копирования байтов более одного раза. Кодеки SBE предоставляют аспект (концепцию) для целей кодирования, декодирования непосредственно из базового буфера и из буфера. Эти ограничения объясняются тем, что сообщения большего размера не поддерживаются напрямую, чем буферы передачи. Протокол фрагментации для сообщений, размер которых превышает предоставленный промежуточный размер, необходим для распространения и настройки большого набора сообщений.

Следующий тип называется *модификацией ваших собственных типов* . Создание бесплатной копии оказывает существенное влияние на процедуру кодирования данных в основном буфере как его типы. Например, 64-разрядное целое число может быть закодировано непосредственно в основном буфере в версии сборки MOV x86_64 общей инструкции на ассемблере . Если процедура байта этого типа определена для другого процессора, байт можно поменять местами в регистре, пока он не будет сохранен в основном буфере с помощью команды BSWAP

x86 . Применение этого принципа позволяет получить доступ к полям в критических языковых языках, таких как C ++ и Java , или для классов и структурированных полей.

Далее, третий тип принципа дизайна - это *свободный выбор*. Следует отметить, что выбор объектов является основой для утечки информации в кэш-памяти процессора и может рассматриваться как снижение производительности. Таким образом, лучше собирать или утилизировать предметы. Использует легковесную модель для разработки свободного выбора кодеков SBE. Окно Flyweight расположено в верхней части основного буфера для прямого шифрования и декодирования данных. Соответствующий тип " flyweight " выбирается внизу заголовка сообщения с идентификатором модели. Внизу заголовка сообщения с идентификатором шаблона находится соответствующий шаблон "Мухи" выбран. Поля данных должны храниться вне границы обработки данных, и в этом случае они должны храниться отдельно.

Четвертое правило - *разрешение потока*. Инновационная память подсистем стала самой сложной. Метод модели доступа позволяет существенно определить эффективность и последовательность использования памяти. Наилучшая эффективность, основанная на дополнительном доступе к файлам и более последовательная задержка достигается путем использования метода потока.

В базовом буфере для кодирования и декодирования данных предусмотрены кодеки SBE на основе пересылки. Следующий тип называется *доступом к одному слову*. Если слова, которые не предназначены для слов, находятся в небольших пределах, многие архитектуры процессоров представляют значительные трудности с производительностью. В этом случае слова имеют начальный адрес, размер которого умножается на байты. Только 4 и тд. 64-битные целые числа, которые делятся на 8 и 32-битные целые числа, начиная с байтовых адресов, которые делятся, и должны поступать только из байтовых адресов. Однако схемы SBE поддерживают «плавающую» теорию, которая устанавливает позицию начального поля в сообщении. Данные кадрирование протокола в 8 байт (границы) на границе инкапсуляции считается. Для выполнения кратких и эффективных сообщений поля должны быть отсортированы в порядке убывания по типу и размеру. И последний тип принципа дизайна - *обратная совместимость*.

Не всегда возможно обновить все системы одновременно в крупных компаниях или через компанию. Для связи, чтобы повысить производительность, форматы сообщений должны быть обратно совместимы, и в этом случае должна существовать более старая система, которая может считывать последнюю версию этого сообщения, и наоборот. На рисунке 2.3 показана система расширений, разработанная в SBE, которая имеет все функции новейших систем, но старая система игнорирует их до тех пор, пока они не будут обновлены, что позволяет добавлять новые дополнительные поля к сообщениям.

Когда необходимо заполнить новые неделимые поля или требуются серьезные структурные изменения, лучше всего использовать последний тип сообщения, поскольку оно не считается семантическим расширением существующего типа сообщения. Сообщения SBE имеют единую тему, которая определяет тип и версию базы сообщений для систематизации. Тема поставляется с поддержкой корневых полей, которые имеют определенную длину с постоянными сдвигами.

Корневые поля очень похожи на синтаксис Си. В этом случае данные считаются наиболее сложным (корневым блоком) корневым блоком, и одна или несколько циклических групп (например, повторяющаяся группа) имеют все возможности для управления. Дублированные группы имеют все возможности для реализации других итеративных групповых схем. Наконец, (строки переменной длины) переменная длина строки доходит до конца сообщения. Кроме того, поля могут не быть обязательными.

Рисунок 2.3 - Структура сообщения SBE

2.3 Развитие научной работы и сбор данных

Отправить данные протоколы, созданные на основе обзора могут быть установлены в некоторых ситуациях для достижения:

1) навыки и опыт работы с устройством необходимы для снижения риска неправильного использования;

2) дизайн клиент-серверного приложения;

3) необходимо ввести данные в базу данных с большим количеством экономических данных;

4) Отправить данные, передаваемые закодированные сообщения в сети , и они должны изучить , как получатель базы данных и хранить;

5) необходимо изучить графические понятия экономических данных.

2.4. Анализ последних применений протокола передачи данных простого двоичного кодирования (SBE)

Каждый день мы сталкиваемся с различными процессами обработки информации с помощью новых технологий. Все операции обмена данными влияют на рост Big Data , который увеличился на 10% по сравнению с прошлым годом. «Сколько данных мы создаем каждый день? Удивительная статистика (How much data Do We Create Every Day? Unbelievable Statistics.)

Подчеркивает, что в последние два года, 90 процентов данных в мире были созданы» [22].

Подавляющее большинство будет составлено предприятиями, а не пользователями Интернета. (Рис. 2.4) [23].

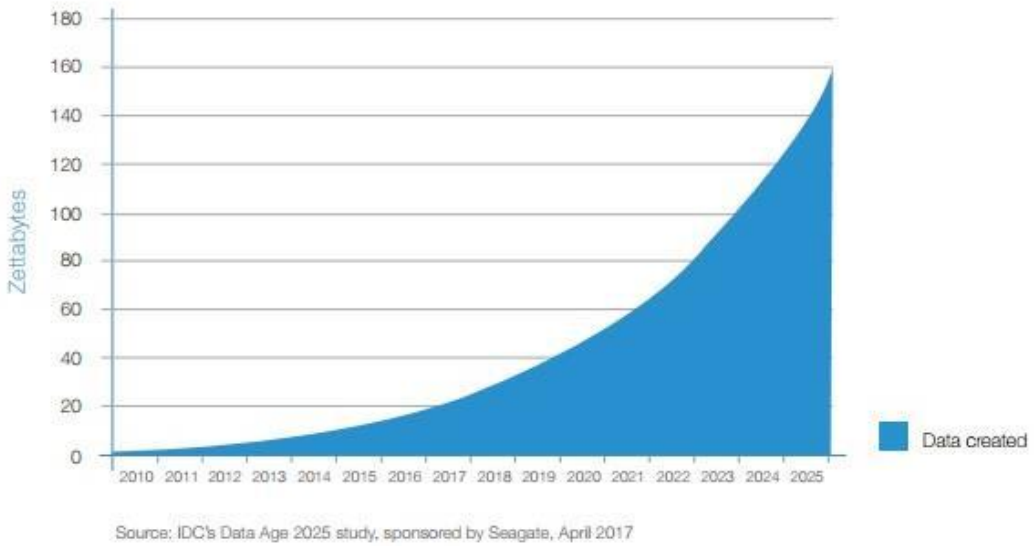


Рисунок 2.4 - Годовой объем глобальной базы данных. Источник: seagate.com

В апреле 2017 года на веб-сайте Seagate был опубликован анализ данных, согласно которому к 2025 году объем данных во всем мире увеличится до 160 зетабайт .

Статистика, опубликованная на портале « Статистика », привлекает внимание к прогнозу роста рынка с 2011 по 2027 год (график 2.5). По данным портала, в 2018 году годовой доход крупнейшего мирового рынка данных ожидается на уровне 42 миллиардов долларов США [24].

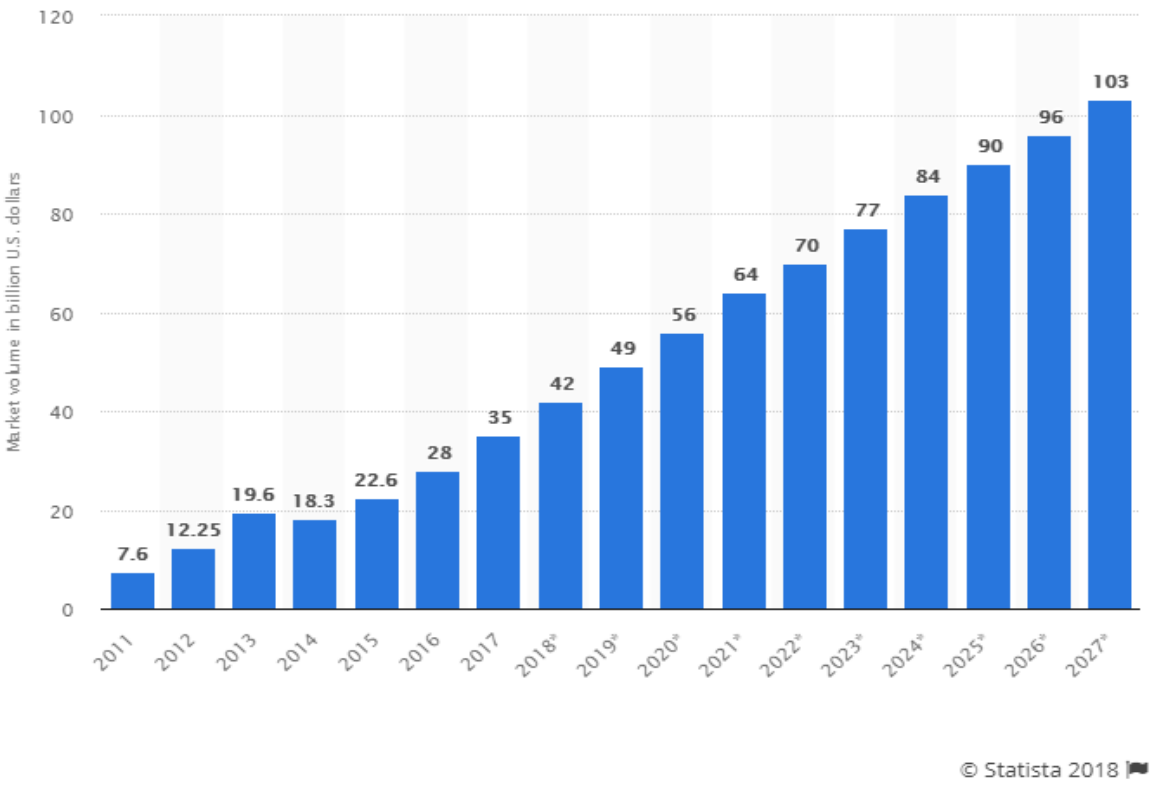


Рисунок 2.5 - Прогноз объема рынка больших данных (в миллиардах долларов США). Источник: statista.com

По сути, строителями финансовых систем являются крупные институты, работающие в сфере торговли, которые часто сопровождаются большим потоком информации при обмене данными. Это потому, что они сталкиваются с проблемой медленной обработки транзакций в повседневной жизни ,

В финансовом мире эту ситуацию называют большой задержкой. Благодаря определенным технологиям большинство компаний пытаются справиться с такими задержками в сети и обеспечивают бесперебойный обмен данными в компьютерных сетях. Такие технологии включают в себя способы передачи данных и протоколы .

Протокол - это набор соглашений, правил или процедур для передачи данных между электронными устройствами, такими как компьютеры. протоколы являются золотым стандартом для финансовой деятельности является. FIX используется в финансовом пространстве как покупателями, так и продавцами.

FIX распространяется из США в Европу и Азию. Протоколы устанавливаются международными или отраслевыми организациями. Они различаются по брокерам, банкам, паевым инвестиционным фондам, биржам и дилерам. Протокол FIX использовался более 25 лет и помог восстановить некорректную систему финансовой торговли за счет введения непрерывного потока информации [20].

Основной особенностью протокола Simple Binary Encoding при обработке и передаче данных является низкая задержка .

В статье «Простое бинарное кодирование для высокой производительности на рынке данных интерфейсов », опубликованная в журнале «Автоматизированное Trader Выпуска 44 Q1 2018», CME Group отмечает, что интерфейсы рыночных данных на Чикагской товарной бирже реализованы с использованием Simple Binary Encoding . удален [21].

Инструмент SBE все еще находится в стадии разработки, но его преимущества были оценены многими. В этой статье говорится: «Стандарт простого двоичного кодирования (SBE) был разработан FIX для балансировки потребностей в задержке и использования полосы пропускания. Поскольку протокол FIX больше не подходит для быстрого обмена, его заменил протокол SBE »[21].

Поэтому протокол SBE решил ускорить как внутренние коммуникации, так и рыночные данные, распространяемые в мире торговли. Проблема заключается в качественной торговле - это техническая проблема для сообщения о задержках, измеряемых в микросекундах и даже наносекундах .

Чтобы найти хорошее решение, которое предлагается на том же уровне, что и SBE, мы определили три основные проблемы протокола FIX, для которых нет альтернативы. Первая проблема заключается в том, что уровень FIX, знакомый тысячам пользователей, остается прежним. Эти требования необходимы протоколу SBE для представления существующих сообщений FIX, а также для их оптимизации для повышения производительности .

Вторая проблема в FIX заключается в том, что кодировка сообщений не подходит для будущих целей. Другими словами, проблема в том, FIX- «s содержит около 20 типов данных, даже если все они , как ASCII текст читабельной кодировке . Чтобы изменить его, такой процесс требует некоторого времени, что является пустой тратой свободного времени на обработку. И последняя проблема с протоколом FIX состоит в том, что некоторые поля пути в кодировке были изменены, то есть высокая изменчивость структуры памяти и большой размер сообщения обычно вызывают ошибки кеша, которые могут останавливать ЦП. Изменчивость, как и FPGA, делает кодирование недопустимым для аппаратных реализаций [21].

Все вышеперечисленные проблемы являются основной причиной создания и использования такого инструмента, как SBE. Создание протокола SBE

Уроки были извлечены из прошлого, и они были оптимизированы для низкой задержки . Задержка - это время, необходимое для отправки пакета из одной точки в другую [25].

Одно из различий между протоколами мгновенной передачи данных заключается в том, что SBE использует свои собственные двоичные типы в проводной сети, которые используются для количественной оценки и количественной оценки других данных. FIX- «ы otızfazalı каждого типа

данных, SBE специфика определяется, двоичное кодирование. Кодирование SBE намного более компактно, чем ASCII-эквиваленты [26].

CME Group , в соответствии с SBE- дата утвержденных достижений.

Поля и типы SBE как константы в метаданных может быть установлен. Нет необходимости посылать «постоянное значение» через сеть, потому что постоянное значение известно как отправителю, так и получателю [26].

Поля могут поддерживаться только битом, который представляет массив значений, таких как «Вкл / Выкл» (или флаги истина / ложь) . Набор битов отправляется на провод в виде целочисленного типа размера (int) от 8 до 64 бит [26].Постоянная память. То есть кодер или декодер сообщений должен проходить через буфер сообщений только один раз [26].

Еще одна хорошая практика - создавать кодеры и декодеры в виде шаблонов в буфере поверх сообщений. Существует одна форма кодера и одна форма декодера для каждого типа сообщений, которые повторяются в течение дня. С SBE память фактически может быть близка к стабильному состоянию [26].

Основные преимущества SBE, отвечающие требованиям интерфейсов электронной коммерции Globex :

- высокое кодирование / декодирование продукта;
- правильное решение использовать пропускную способность;
- стандартизированный тип поля;
- определение стандартизированных сообщений;
- гибкая структура сообщений;
- прямой доступ к данным;
- полная интеграция с протоколом FIX [26].

3 СОЗДАНИЕ ПРОГРАММЫ

3.1 Системная архитектура

Многие клиенты (удаленные процессоры) запрашивают и получают услуги от централизованного сервера (хост-компьютера). Вы также можете открыть и закрыть сокет, чтобы освободить порт от передачи информации . В связи с этим клиентские компьютеры предоставляют интерфейс, который позволяет пользователю компьютера запрашивать серверные услуги и отображать результаты возврата сервера. Этот процесс описывает архитектуру клиент-сервер. Поэтому была построена архитектура проекта мгновенной передачи данных (рисунок 3.1).

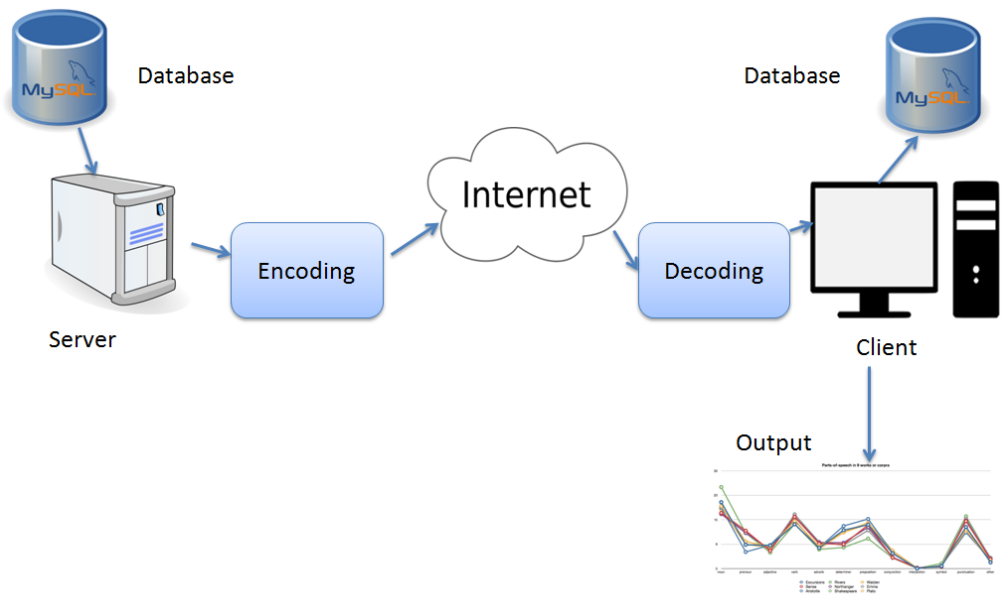


Рисунок 3.1 - Архитектура проекта

Основной процесс начинается с того, что сервер ожидает, пока клиент подтвердит соединения. После положительного ответа сервер извлекает финансовые данные из базы данных, кодирует их с помощью простого двоичного кодирования и отправляет их клиенту. Клиент, который получает большое количество информации, декодирует ее и сохраняет в базе данных. Кроме того, предоставляется графическая сводка финансовых данных.

Для реализации программного продукта был выбран язык программирования - Java . Java - это объектно-ориентированный язык программирования высокого уровня, а также программная платформа. Этот язык программирования упрощает сетевое программирование благодаря специальным инструментам и классам. Передача данных через клиент-серверные приложения основана на использовании протоколов приложений TCP / IP. СУБД MySQL выбрана для хранения , поддерживает несколько запросов одновременно, несколько концентраций, а также фиксированные и переменные

Есть ряд преимуществ, которые включают запись длины.

3.2 Работа сервера

Чтобы подключить сервер к клиенту, была выбрана библиотека Java New I / O для высокопроизводительных операций ввода-вывода. Это также позволяет быстро и легко разрабатывать сетевые приложения, такие как серверы и клиентские протоколы. Это упрощает и ускоряет сетевое программирование, такое как серверы сокетов TCP и

UDP . Исследование этой библиотеки выявило ряд преимуществ перед вводом- выводом Java (см. Таблицу 3.1).

Таблица 3.1 - Основные различия между библиотеками

Java IO	Java NIO
Flow-ориентированный	буфер-ориентированный
Блокировка ввода / вывода (I / O)	Разблокированный ввод- вывод
	Селекторы

Буферно-ориентированный подход Java NIO отличается тем, что данные читаются в буфере, который затем обрабатывается. Вы можете перемещаться вперед и назад в буфере столько, сколько нам нужно. Это дает некоторую гибкость при обработке. Однако, если буфер содержит все данные, он должен быть полностью обработан. При чтении больших данных в буфере убедитесь, что данные в буфере не обработаны. Неблокирующий режим Java NIO позволяет потокам запрашивать данные для считывания с канала и получать только то, что доступно или не доступно вообще, если в настоящее время нет данных. Кроме того, использование библиотеки NIO улучшает пропускную способность, низкую задержку и низкое потребление ресурсов. На рисунке 3.3 показано описание метода open () в серверном отсеке .

Основные компоненты NIO: селектор и selectionKey , каналы, буферы . Объект класса Selector-Selector . Селектор для каждого экземпляра сокета способен контролировать большое количество каналов и, следовательно, множество подключений. Интересно, что когда вы находитесь на канале (например, клиент пытается установить связь или выполняются операции чтения и записи), селектор уведомляет приложение для обработки запроса.

Селектор делает это путем создания ключей, которые являются экземплярами класса SelectionKey [27]. Каждый ключ содержит информацию о запросе и типе этого запроса. Тип может быть одним из следующих: операция соединения (клиентом), прием соединения (сервер), операция чтения, операция записи. Приложение А описывает использование SelectionKey и условия процесса для каждого типа.

Сокетный канал - это экземпляр нового класса сокетного канала, который позволяет передавать данные по сети. Метод open () создает экземпляр сокет-канала configureBlocking (false) устанавливает канал как разблокированный. Подключение к серверу осуществляется методом подключения. IP-адрес сервера и порт связи также отображаются. Создать селектор открытый () селектор будет вызывать класс. Наконец, метод register подключает селектор канала сокета .

Используя OP_ACCEPT, селектор только сообщает, что клиент пытается установить соединение с сервером. Другими возможными вариантами являются OP_CONNECT, OP_READ и OP_WRITE, используемые клиентом.

```java
public void open(int port){
    if(DEBUG)System.out.println("araylim.server.Server.open()");
    if(selector != null) return;
    if(serverSocketChannel != null) return;

    try{
        Class.forName("com.mysql.jdbc.Driver");
        connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/forex","root","root");
        statement = connection.createStatement();



        selector = Selector.open();
        serverSocketChannel = ServerSocketChannel.open();
        serverSocketChannel.configureBlocking(false);
        serverSocketChannel.socket().bind(new InetSocketAddress(port));

        serverSocketChannel.register(selector, SelectionKey.OP_ACCEPT);

        Platform.runLater(() -> {
            logs.add("server started");
        });
    }catch(Exception e){
        Platform.runLater(() -> {
            logs.add("error : couldn't start server");
        });
    }
}
```

Рисунок 3.3 - Соединение с клиентом

Следует отметить, что селектор ожидает события, когда клиент пытается установить контакт. В это время серверные приложения получают селектор и ключи, созданные для каждого ключа, который проверяет его тип. Вы можете заметить, что ключ был обработан, его необходимо удалить из набора ключей, вызвав метод remove . Если тип ключа является приемлемым (isAcceptable () = true), сервер находит клиентский канал , вызывая метод accept, и устанавливает его как разблокированный , таким образом записывая его в селектор с помощью OP_READ . Также OP_WRITE или OP_READ | Вы можете использовать OP_WRITE варианты .

3.3 Структура данных SBE

Финансовые данные были изучены для заполнения базы данных. Для выполнения SQL-запросов использовался соединитель драйвера JDBC, представляющий собой интерфейс прикладной программы, состоящий из множества классов и интерфейсов, написанных на Java . База данных заполнена историческими данными Форекс [27]. Форекс - это международный финансовый рынок, на котором валюта обменивается. База данных имеет 6 атрибутов: открытый , высокий , низкий , закрытый , дата и время . OHCL сессия открывается, цена закрытия во время сессии , жира и самой низкой цене, а также M1

Таймфрейм описывает дату и время процесса (одна минута) сеанса. Количество данных в базе данных составляет 5049687. Структура не сложная, но финансовых данных много. Данные были заполнены с очень высокой скоростью с помощью команды LOAD DATA INFILE, которая считывает путь из текстового файла в таблице.

3.4 Описание схемы SBE

Простое двоичное кодирование (SBE) - это метод кодирования и декодирования сообщений в двоичном формате на основе схем. Сообщения - это единица информации для обмена данными между системами на основе протоколов FIX / SBE. SBE использует формат XML для определения сообщений, заголовков и других элементов. [28].

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<sbe:messageSchema xmlns:sbe="http://www.fixprotocol.org/ns/simple/RC3"
                   package="araylim.sbe"
                   id="1"
                   version="0"
                   semanticVersion="5.2"
                   description="schema"
                   byteOrder="littleEndian">
    <types>
        <composite name="messageHeader" description="Message identifiers and length of message root">
            <type name="blockLength" primitiveType="uint16"/>
            <type name="templateId" primitiveType="uint16"/>
            <type name="schemaId" primitiveType="uint16"/>
            <type name="version" primitiveType="uint16"/>
        </composite>
    </types>

    <sbe:message name="forexData" id="1" description="description of forex data">
        <field name="datetime"  id="1" type="uint64"/>
        <field name="open"      id="2" type="float"/>
        <field name="high"      id="3" type="float"/>
        <field name="low"       id="4" type="float"/>
        <field name="close"     id="5" type="float"/>
    </sbe:message>
</sbe:messageSchema>
```

Рисунок 3.4 - Описание схемы SBE

Система набрана и пользователи могут создавать новые типы. Рисунок 3.4 показывает схему сообщения для финансовых данных .

Окончательный документ в формате XML является элементом messageSchema. Этот элемент имеет следующие атрибуты, например: package , ID, version, semanticVersion, description и byteOrder [29].

Пакет - это пространство имен для C ++ и имена пакетов для Java . ID является уникальным идентификатором для схемы. Версия - это номер версии зарегистрированных сообщений и типов, который по умолчанию равен нулю. SemanticVersion - семантическая версия информации в виде строки. Описание

- описание всех схем сообщений . ByteOrder - это количество байтов по умолчанию для всех типов сообщений (bigEndian или LittleEndian) [30].

Кодирование и декодирование сообщений для SBE фокусируется на наборе основных простых типов. Он включает в себя: int8 (8-битное кодирование), int16 (16-битное кодирование), int64 (64-битное кодирование) и многое другое.

Структура закодированных типов является Составной . композитный

- кодирование в порядке публикации. Некоторые составные элементы используются для объявления заголовка сообщения, а также для кодирования переменных длин строк и кодирования размера группы .

Элемент < Composite > имеет атрибуты: имя , описание .

Дублирующиеся группы, а также данные переменной длины находятся в поле элемента сообщения . Процедуры < Field >, < group > и < data > помещают данные в каждое сообщение. Полный оператор < Field > должен превышать < group > и < data >. Этот параметр гарантирует, что все поля фиксированной длины находятся перед сообщением. Данные с переменной длиной сохраняются после дублирования групп. Элемент < Message > имеет атрибуты: имя , описание , идентификатор.

Элемент поля (Field) указывает размер поля сообщения. Имя каждого атрибута записывается в элементе < name >. Эти сообщения содержат пять из них: datetime , open , high , low , close . Удобно, что есть идентификатор сообщения (id) и описание типа (типа). Uin64 (64-разрядное неизвестное целое число) было выбрано для DateTime . Для других это тип float [31].

SBE инструментом является утилитой командной строки , используемой для создания кодеков и тестовых шаблонов сообщений. Java tälände инструмент и может работать как исполняемый JAR - файл [31]. Схема сообщения может быть встроена в инструмент SBE и предназначена для блокировки источников на разных языках, в нашем случае это язык программирования Java, и предоставляет ему следующую команду:

Java [- Doption = значение] - банка sbe.jar <сообщение-заявления-file.xml>

Сообщения принимаются к отправке в кадрирования протокола. Кадр определяет размер буфера, содержащего заголовок сообщения и само сообщение (см. Рисунок 3.5).

```
+-------------------------------------------------------------+
|        +----------+-----------------------------------+      |
|Frame   |Msg Header|Message Body                       |      |
|        +----------+-----------------------------------+      |
+-------------------------------------------------------------+
```

Рисунок 3.5 - Фрейм сообщения

Кадр может содержать поля сеанса или уровня передачи, которые принадлежат разным уровням модели

OSI и исключаются из кодеков сообщений, относящихся к шестому уровню модели OSI.

3.5 Кодирование и декодирование данных

Схема протокола SBE в качестве инструмента ввода, а затем он bitewişterge (заглушки) генерировать соответствующий язык. Пломбы используются для кодирования и декодирования сообщений из буфера финансовых данных.

В начале реализации протокола мы берем блокирующие файлы, которые создаются, и используем их в программе клиент-сервер. Сервер ожидает соединения с клиентом, если клиент подтверждает соединение, он берет данные из базы данных, кодирует их с помощью SBE и отправляет их клиенту. Для процесса кодирования сообщений вы должны сначала закодировать тему, а затем выполнить процесс извлечения финансовых данных из базы данных.

```
while(rs.next()){

    ByteBuffer byteBuffer = ByteBuffer.allocateDirect(64);
    UnsafeBuffer directBuffer = new UnsafeBuffer(byteBuffer);
    int bufferOffset = 0;

    MESSAGE_HEADER_ENCODER
        .wrap(directBuffer, bufferOffset)
        .blockLength(FOREX_DATA_ENCODER.sbeBlockLength())
        .templateId(FOREX_DATA_ENCODER.sbeTemplateId())
        .schemaId(FOREX_DATA_ENCODER.sbeSchemaId())
        .version(FOREX_DATA_ENCODER.sbeSchemaVersion());

    bufferOffset += MESSAGE_HEADER_ENCODER.encodedLength();

    int id  = rs.getInt("id");
    Date date = rs.getDate("date");
    Time time = rs.getTime("time");
    float open = rs.getFloat("open");
    float high = rs.getFloat("high");
    float low = rs.getFloat("low");
    float close = rs.getFloat("close");
    int volume = rs.getInt("volume");

    LocalDate datePart = new LocalDate(date);
    LocalTime timePart = new LocalTime(time);
    DateTime dateTime = datePart.toDateTime(timePart);

    long datetime = dateTime.getMillis();
```

Рисунок 3.6 - Кодирование темы сообщения

Заголовок сообщения содержит несколько полей. Он декодирует, какой кодек используется в качестве шаблона для сообщения. TemplateId - это идентификатор типа шаблона сообщения. Длина блока - это длина корневых сообщений блока перед повторением групп или переменных данных. Версия - это версия схемы, которая сообщает расширения , которые действительны для схемы (рисунок 3.6).

Далее мы кодируем базу сообщения (см. Рисунок 3.7):

```
FOREX_DATA_ENCODER
        .wrap(directBuffer, bufferOffset)
        .id(id)
        .datetime(datetime)
        .open(open)
        .high(high)
        .low(low)
        .close(close)
        .volume(volume);


ByteBuffer source = directBuffer.byteBuffer();

dataTracking.put(channel, source);
key.interestOps(SelectionKey.OP_WRITE);
currentRow++;
```

Рисунок 3.7 - Кодирование базы сообщения

SBETool.java включает в себя интерфейс Java для преобразования буфера кодирования . SBE также работает с байтовым буфером, используя класс DirectBuffer, и предлагает абстракцию для Java [31]. Герметик оборачивается вокруг буфера и последовательно читает его с самого начала. После извлечения данных из базы данных сначала необходимо закодировать имя субъекта , а затем сообщение.

После подтверждения клиент получает данные в буфере, кодирует финансовые данные и сохраняет их в базе данных. В процессе декодирования вам необходимо декодировать тему, а затем искать шаблон, который можно использовать для декодирования базы сообщения. Кроме того, сохраненные данные могут быть представлены в световой диаграмме и гистограмме через библиотеку JavaFX .

Процесс декодирования с использованием SbeTool :

Сначала мы декодируем тему сообщения , а затем в базе данных могут храниться финансовые данные и фон (см. Рисунки 3.8 и 3.9).

```
UnsafeBuffer directBuffer = new UnsafeBuffer(readBuffer);

int bufferOffset = 0;
MESSAGE_HEADER_DECODER.wrap(directBuffer, bufferOffset);
int templateId = MESSAGE_HEADER_DECODER.templateId();
int actingBlockLength = MESSAGE_HEADER_DECODER.blockLength();
int schemaId = MESSAGE_HEADER_DECODER.schemaId();
int actingVersion = MESSAGE_HEADER_DECODER.version();

bufferOffset += MESSAGE_HEADER_DECODER.encodedLength();
```

Рисунок 3.8 - Расшифровка темы сообщения

Затем мы храним финансовые данные в базе данных (рисунки 3.11 и 3.12).

```
StringBuilder sb = new StringBuilder();
FOREX_DATA_DECODER.wrap(directBuffer, bufferOffset, actingBlockLength, actingVersion);

try{


    Date date = new Date(FOREX_DATA_DECODER.datetime());
    Time time = new Time(FOREX_DATA_DECODER.datetime());

    preparedStatement.setLong(1,FOREX_DATA_DECODER.id());
    preparedStatement.setDate(2,date);
    preparedStatement.setTime(3,time);
    preparedStatement.setFloat(4,FOREX_DATA_DECODER.open());
    preparedStatement.setFloat(5,FOREX_DATA_DECODER.high());
    preparedStatement.setFloat(6,FOREX_DATA_DECODER.low());
    preparedStatement.setFloat(7,FOREX_DATA_DECODER.close());
    preparedStatement.setLong(8,FOREX_DATA_DECODER.volume());

    preparedStatement.execute();
}catch(Exception e){
```

Рисунок 3.9 – Расшифровка базы сообщений и сохранение ее в базе данных.

Как только хранение данных в базе данных завершено, клиент предоставляет графическую сводку финансовых данных. Вывод данных делится на два: в виде линейной таблицы и в виде японских огней. Библиотека JavaFX была использована для реализации . Это набор графических и мультимедийных пакетов, которые позволяют разработчикам проектировать, создавать, тестировать, настраивать и развертывать приложения с расширенными возможностями для клиентов, которые запускаются последовательно для разных платформ. На рисунке 3.10 показан линейный график финансовых данных:

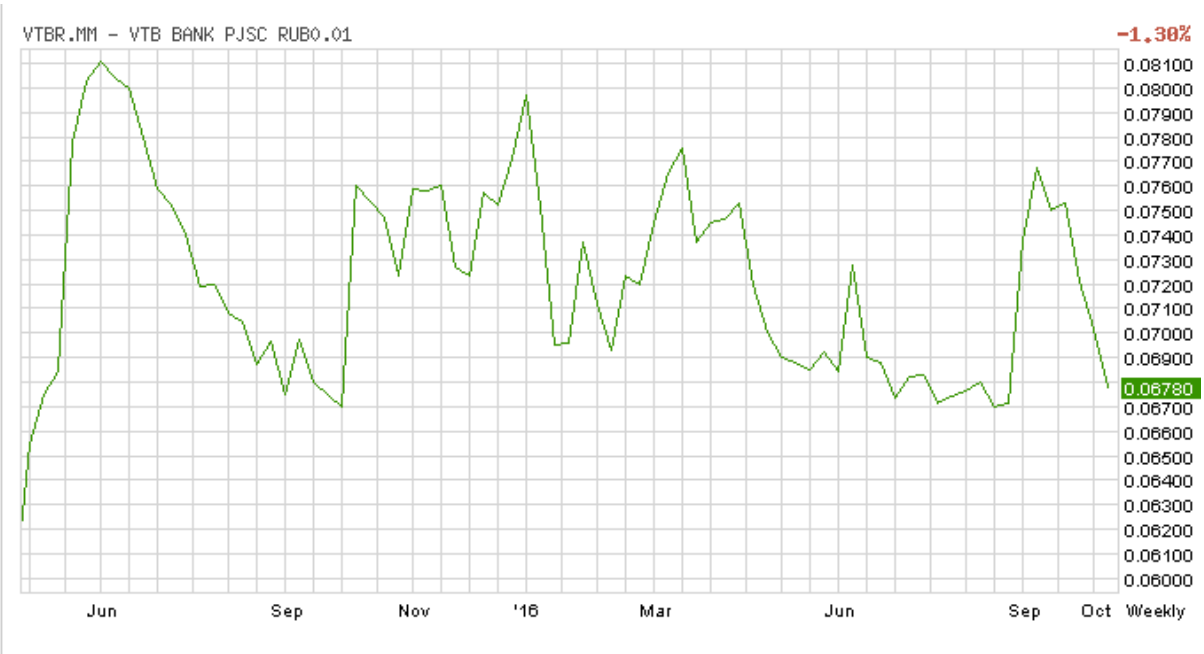


Рисунок 3.10 - Вывод линейных графиков

Ось X - это время, а ось Y - цена закрытия. Далее, на рисунке 3.11 показан график в форме японской лампы, которая имеет все четыре атрибута (открытие, максимум, минимум, закрытие):

Рисунок 3.11 - Выпуск в виде японской лампы

Широкий корпус лампы представляет цену открытия и закрытия в течение определенного периода времени, а фитиль - верхняя и нижняя линия лампы - максимальная и минимальная цена за этот период. Японские огни могут отображать реальные данные за определенный период времени.

3.6 Биржа и обзор официального сайта

Bitstamp является одной из немногих бирж, которые могут приносить деньги с помощью традиционных способов оплаты, таких как банковские карты. Таким образом, вы можете использовать Bitstamp для покупки биткойнов, а затем управлять ими по своему желанию.

Обмен принадлежит Bitstamp Ltd. Имеет офисы в Люксембурге, Лондоне и Нью-Йорке . Bitstamp - первая полностью лицензированная в Европе крипто- вальюта Exchange. Он регулируется Люксембургской комиссией по финансовому надзору (CSSF). Этот факт эффективно отделяет Bitstamp от других бирж , большинство из которых не регулируются, что создает определенные риски. Лицензия CSSF гарантирует, что компания периодически подотчетна ответственным службам. Bitstamp зарегистрирован как «платежный орган», поэтому вы всегда можете отправлять и получать деньги на бирже .

Bitstamp был взломан в начале 2015 года после интернет-атаки. После этого компания полностью переработала свою торговую платформу. Они утверждают, что 98% средств клиентов хранятся в холодильных камерах, и только 2% доступны непосредственно на бирже. Это значительно улучшило систему безопасности .

В июле 2017 года Bitstamp подписал соглашение со Swissquote. Швейцарский банк и брокер решили довериться бирже, чтобы позволить своим клиентам торговать биткойнами . Это говорит о надежности Bitstamp. Банк не интегрировал торговую систему Биткойн с платформой MetaTrader4 (MT4). Все торговые операции осуществляются на отдельной площадке банка с ограниченной ответственностью .

Отзывы пользователей Bitstamp неоднозначны. Большинство людей удовлетворены обслуживанием, но есть также люди, которые не удовлетворены работой службы поддержки. Как Poloniex, Bitstamp пользователи , вероятно , не готовы плавать.

Bitstamp против криптовалюты EUR и USD позволяет продавать. К сожалению, на многих других биржах, таких как криптовалюта платформы не так много. В дополнение к биткойнам Bitstamp вы можете найти пары Litecoin и Ripple . Если вы хотите получить другой интересующий вас код, из Bitstamp, эти три криптографа - вы можете использовать его для обмена одной валюты, тогда они заинтересованы в криптографической поддержке, которую вы можете сосредоточить на обмене.

Bitstamp не поддерживает маржинальную торговлю.

Комиссия за конвертацию Fiat в Bitstamp достигает 0,25%. Как и многие другие биржи, чем выше объем транзакций, тем ниже комиссия. Bitstamp (вместо того, чтобы купить другие вещи агрессивных трейдеров разместили заказы на) скидки для разных людей. Множество крипто-квадратов валютных бирж, предоставляющих трейдерам привилегии ликвидности. В то же время сумма комиссий полностью соответствует средней рыночной ситуации. На приведенном ниже рисунке 3.12 Bitstamp. Обмен данными кэша https:www.bitstamp.net/api/transactions/ ссылку выше.

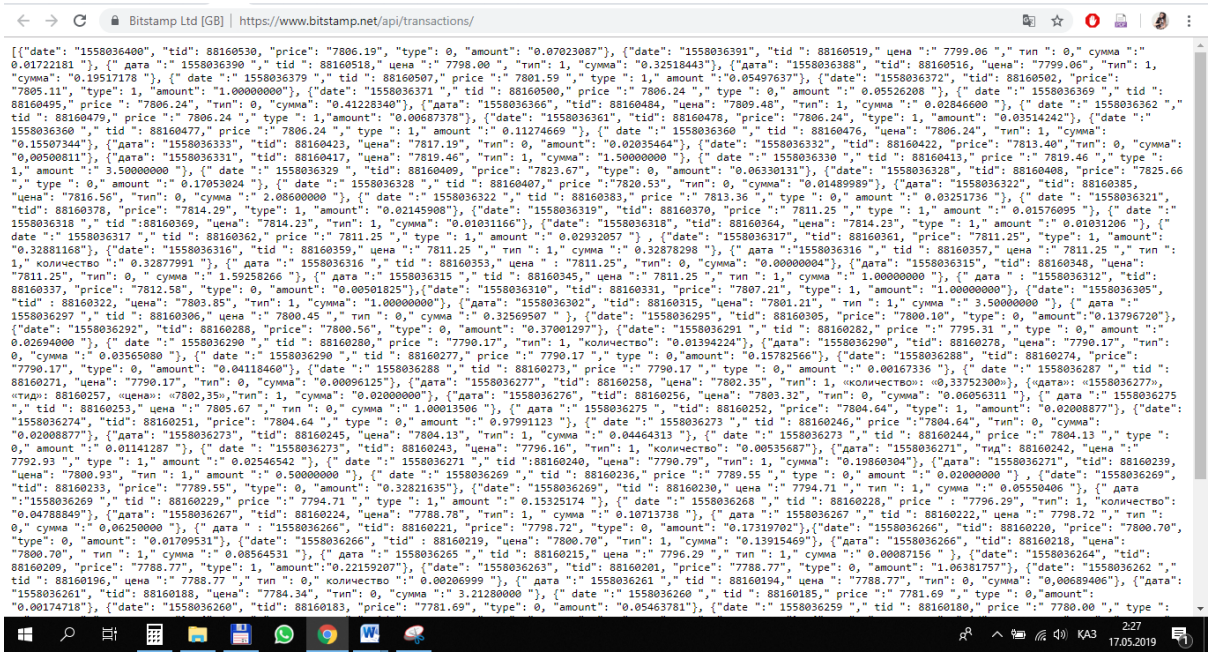


Рисунок 3.12 - Данные Bitstamp

Доступ к платформе Bitstamp осуществляется через веб-интерфейс. В обращении удобно, упор делается на составление графиков. В этом отношении Bitstamp похож на платформу cTrader , но не обладает такой богатой функциональностью. Многие технические индикаторы и инструменты были реализованы на платформе. Таким образом, Bitstamp обходит многих конкурентов по этому параметру .

Bitstamp принимает банковские переводы, банковские карты и валюту криптона. Таким образом, очень легко выставить указ на бирже, что, вероятно, является его главным преимуществом.

Bitstamp - первая в Европе кастомная крипто- вальюта Exchange. Компания хранит 98% пользовательских средств в холодных кошельках, что означает высокий уровень безопасности. Bitstamp является одной из немногих бирж, которые принимают банковские карты и переводы. Это позволяет продавать три основные криптовалюты против евро и доллара США . На самом деле, Bitstamp криптовалютапрекрасная возможность войти в мир, но продажа не представляется возможным.

На рисунках 3.13 и 3.14 мы рассматриваем диаграммы, созданные программой. Эти графики показывают рост и падение продажной цены в настоящее время развивающейся платежной системы баланса Биткойн (BTC) в различных интернет-сетях . Bitstamp является одним из немногих типов обменов, которые могут приносить деньги с помощью традиционных способов оплаты, таких как банковские карты. В картине на Bitstamp Bitcoin Доллар (USD) обменный курс состоянии.

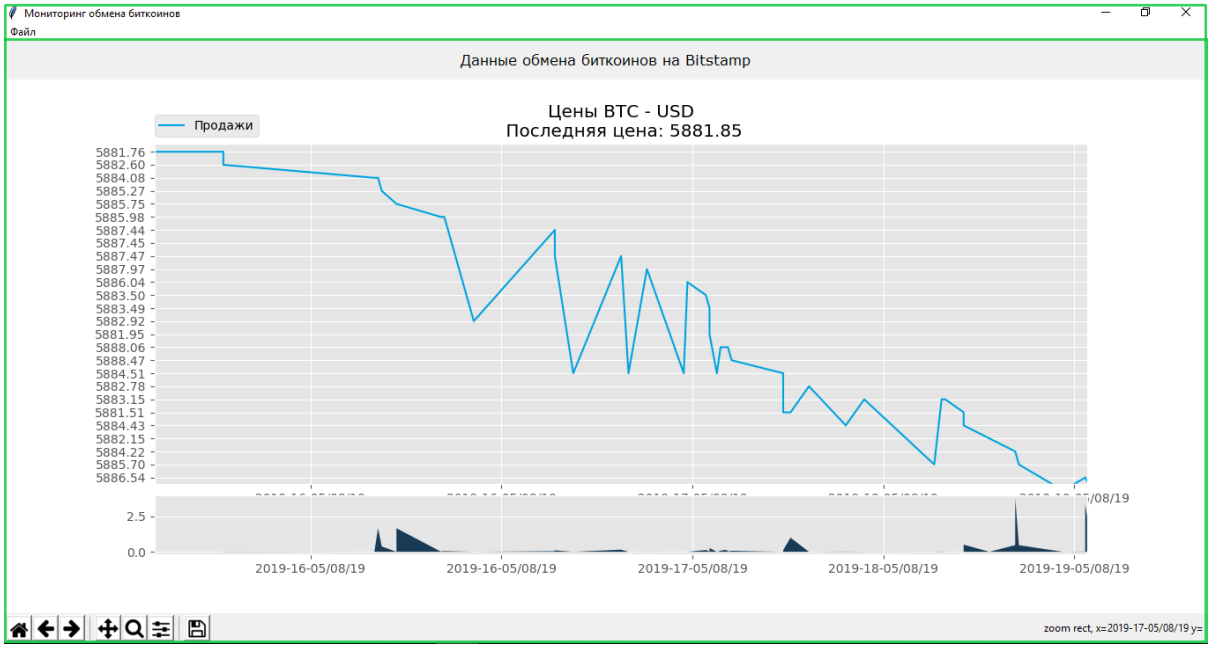


Рисунок 3.13 - Диаграмма состояния биткойнов на бирже Bitstamp

Рисунок 3.14 - Диаграмма статуса японской биткойны на бирже Bitstamp

На следующем рисунке показано состояние нескольких бирж на разных диаграммах (рисунок 3.15).

Рисунок 3.15 - Состояние бирж на разных графиках

3.7 Сравнительный анализ SBE и JSON

В настоящее время не существует протокола, который является полным аналогом простого двоичного кодирования . Но есть аналогичные протоколы для отправки данных. Метод передачи данных JSON был выбран для сравнительного анализа. Это один из самых удобных форматов данных при взаимодействии с JavaScript . Используется, когда вам нужно получить данные с сервера и отправить их клиентскому устройству . Таким образом, промежуточный формат - JSON считается наиболее удобным и распространенным для передачи по сети.

Эти же финансовые данные использовались для отслеживания скорости передачи данных для сериализации . Структура базы данных не сложна, но в ней много данных. Учитывая, что SBE предназначен для более сложных структур, мне было интересно его взаимодействие с базами данных простых структур данных.

Далее описывается сериализация и десерификация финансовых данных и их взаимодействие с базами данных:

Server side:

```
JsonObject json = new JsonObject(); json.addProperty("id", id);
json.addProperty("datetime", datetime); json.addProperty("open",open);
json.addProperty("high", high); json.addProperty("low", low);
json.addProperty("close", close); json.addProperty("volume", volume);

ByteBuffer source = ByteBuffer.wrap(json.toString().getBytes());
```

Client side:

```
JsonParser parser = new JsonParser();
JsonObject json = parser.parse(data_string.trim()).getAsJsonObject();
```

Save data in database:

```
Date date = new Date(json.get("datetime").getAsLong()); Time time =
new Time(json.get("datetime").getAsLong());

preparedStatement.setLong(1,json.get("id").getAsLong());
preparedStatement.setDate(2,date); preparedStatement.setTime(3,time);
preparedStatement.setFloat(4,json.get("open").getAsFloat());
preparedStatement.setFloat(5,json.get("high").getAsFloat());
preparedStatement.setFloat(6,json.get("low").getAsFloat());
preparedStatement.setFloat(7,json.get("close").getAsFloat());
preparedStatement.setLong(8,json.get("volume").getAsLong());
preparedStatement.execute();
```

Мы также хотели бы отметить, что существуют различия между кодированием и сериализацией . Сериализация - это процесс преобразования любого объекта в строку, а десериализация преобразует эту строку в ее производную форму. А кодирование относится к аспектам безопасности, защите данных, преобразованию данных в кодированный формат, который никто не понимает, кроме декодирования , с использованием алгоритма для кодирования данных. Несмотря на простую структуру SBE JSON- показал , что больше по сравнению с шифрованием данных и скорости кодирования.

Результаты можно увидеть в таблице 3.2 и на рисунке 3.16:

Таблица 3.2 - результаты сравнительного анализа

Число директивов	SBE	JSON
1	~30000ns	~40000ns
1000	~271ms	~392 ms
10000	~1998ms	~2412ms

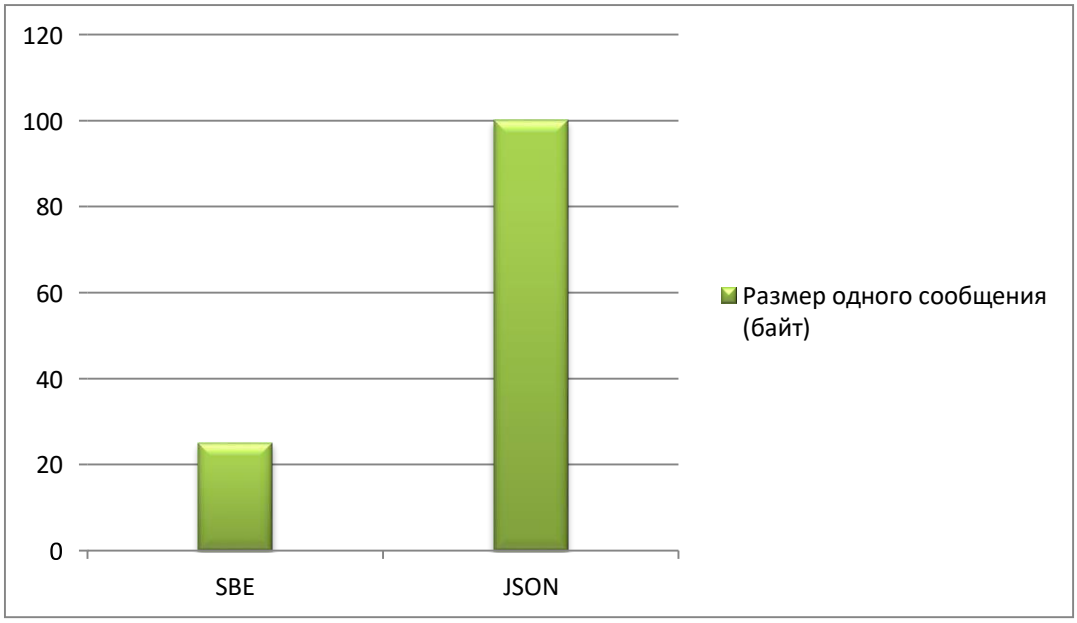


Рисунок 3.16 - Размер одного сообщения при отправке

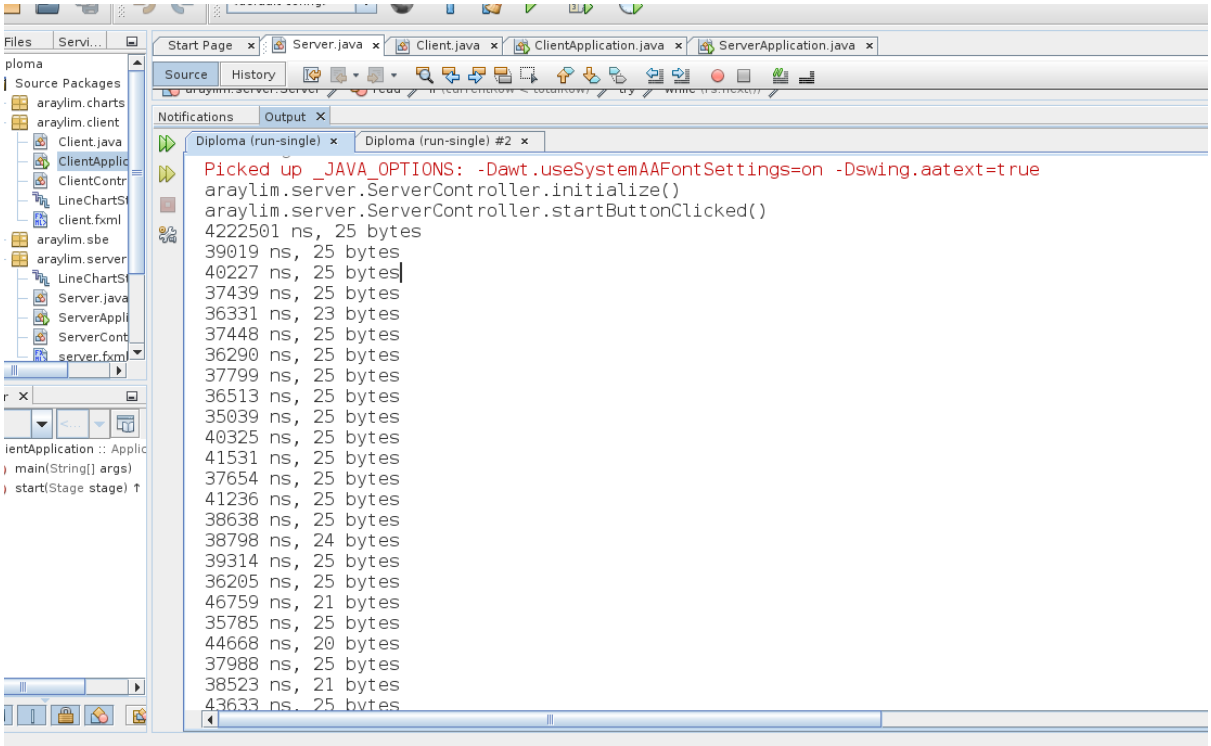


Рисунок 3.17 - Размер сообщения SBE

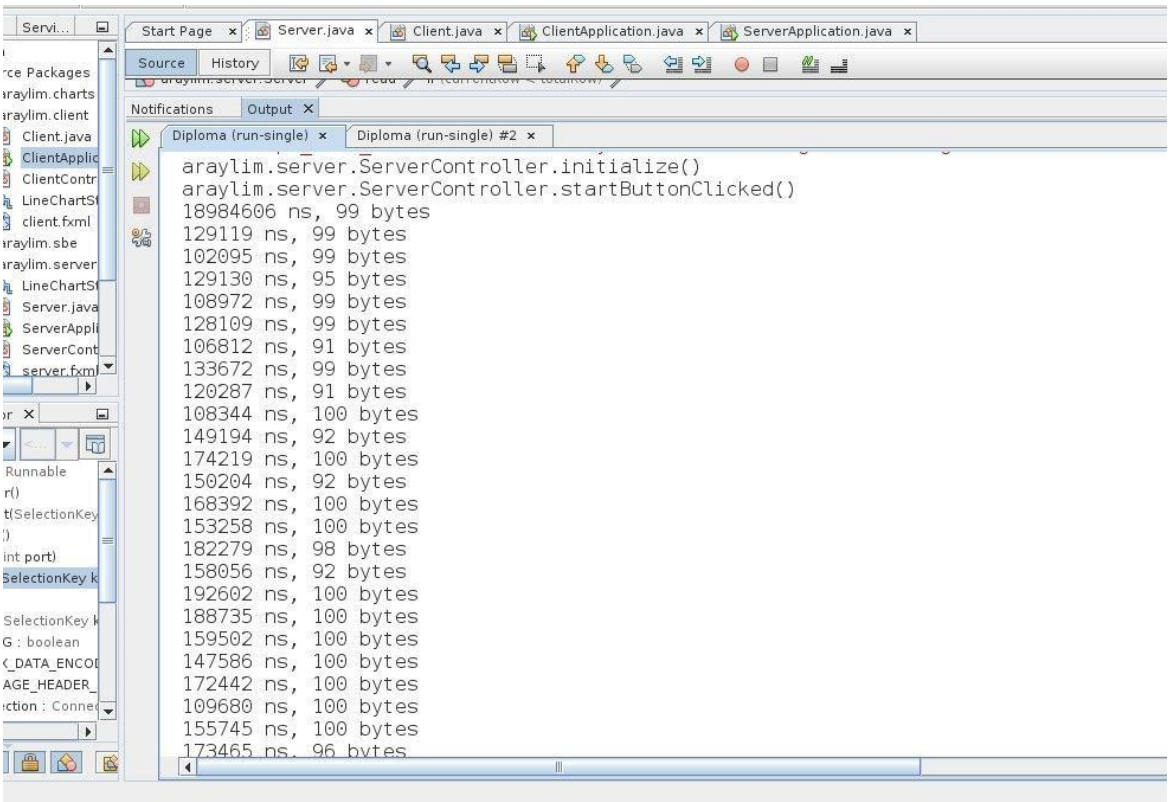


Рисунок 3.18 - Размер сообщения JSON

Результаты показывают, что протокол SBE быстрее, чем протокол JSON. Кроме того, размер одного сообщения, отправляемого с сервера клиенту, менее чем в 4 раза превышает размер сообщения SBE из JSON.

На рисунках 3.17 и 3.18 показаны результаты измерений для каждого сообщения, отправленного через SBE и JSON.

ЗАКЛЮЧЕНИЕ

Результатом данной работы стала интеграция методов сериализации и класстеризации данных для обработки и передачи больших объемов финансовой информации с использованием протокола SBE. Основной процесс начинается с того, что сервер ожидает, пока клиент подтвердит соединение. Когда получен положительный ответ, сервер получает финансовые данные из базы данных, кодирует их с помощью простого двоичного кодирования и отправляет их клиенту. Клиент, который получает большое количество информации, декодирует ее и сохраняет в базе данных. Результаты сравнительного анализа показали, что использование простого двоичного кодирования в Simple Binary Encoding увеличивает скорость и обрабатывает большие объемы финансовых данных.

По результатам исследования было проведено сравнение протоколов SBE и JSON для быстрой отправки финансовой информации, используемой на биржах. Результаты показали, что протокол SBE быстрее, чем JSON. Также при копировании с сервера на клиент для SBE размер одного сообщения в 4 раза меньше, чем JSON. Результаты сравнительного анализа показали, что использование простого двоичного кодирования направлено на увеличение скорости передачи и обработки больших объемов финансовых данных.

Данные от обмена криптовалюты Bitstamp были взяты в виде линейных графиков и японских индикаторов на интерфейсе, который мог бы отслеживать состояние текущих данных, то есть обменный курс биткойнов. Полученные результаты были успешно использованы для выполнения других заданий по теме магистерской диссертации.

СПИСОК ЛИТЕРАТУРЫ

1 Forex Historical Data, BestWebSoft, 2013. http://www.histdata.com/download-free-forex-historical-data/?/metatrader/1-minute-bar-quotes/EURUSDo

2 Томсон. М. Real-logic/simple-binary-encoding, 2014. https://github.com/real-logic/simple-binary-encoding/wiki/FIX-SBE-XML-Primer

3 Томсон. М. Mechanical Sympathy, 2014. http://mechanical-sympathy.blogspot.com/2014/05/simple-binary-encoding.html

4 ITInvest - Способы передачи финансовых данных: протокол FAST, 2014. https://habrahabr.ru/company/itinvest/blog/243657/

5 Google Developers - Protoloc Buffers, 2015. https://developers.google.com/protocol-buffers/

6 Умурзаков Б., Кашкынбек И., Data link system via Financial Information Exchange protocol using Simple Binary Encoding - Indian Journal of Computer Science and Engineering, 2015, том 6, вып. 2, с 60-62.

7 Что такое Big Data: классификация и примеры., 2018. https://neurohive.io/ru/osnovy-data-science/big-data/

8 Awanish. «Big Data Tutorial: All You Need To Know About Big Data!», 2010.

9 Margaret Rouse. «Big data tutorial: Everything you need to know.», 2018.

10 Marshall Cline. «C++ FAQ: «What's this «serialization» thing all about?», 2015.

11 Способы передачи финансовых данных #2: протокол FAST https://habr.com/en/company/iticapital/blog/243657/

12 Сериализация, сэр! Сегодня на ужин байтовая каша, сваренная из объектов C++ https://habr.com/ru/company/xakep/blog/258959/

13 Protobuf benchmarks do not use protobufs optimally https://github.com/real-logic/simple-binary-encoding/issues/128

14 Compare SBE and Protobuf's popularity and activity https://java.libhunt.com/compare-simple-binary-encoding-vs-protobuf

15 Mechanical sympathy: Simplu Binary Encoding https://mechanical-sympathy.blogspot.com/2014/05/simple-binary-encoding.html

16 Проблемы и перспективы рынка финансовых данных: Big Data, Smart Data, большие персональные данные https://www.youtube.com/watch?v=zHZrL9fAZtE

17 D. Mendelson, F. Malabre, Simple Binary Encoding For High Performance Market Data Interfaces, 2018.

18 A Look Into FIX Protocols,

19 https://medium.com/xtrd/a-look-into-fix-protocols-72ec15868e65

20 B. Marr., How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read., 2018.

21 https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we- create-every-day-the-mind-blowing-stats-everyone-should-read/#acdaf660ba99

22 Data Age 2025: The Evolution of Data to Life-Critical Don't Focus on Big Data; Focus on the Data That's Big., 2017.

23 https://www.seagate.com/www-content/our-story/trends/files/Seagate-WP-DataAge2025-March-2017.pdf

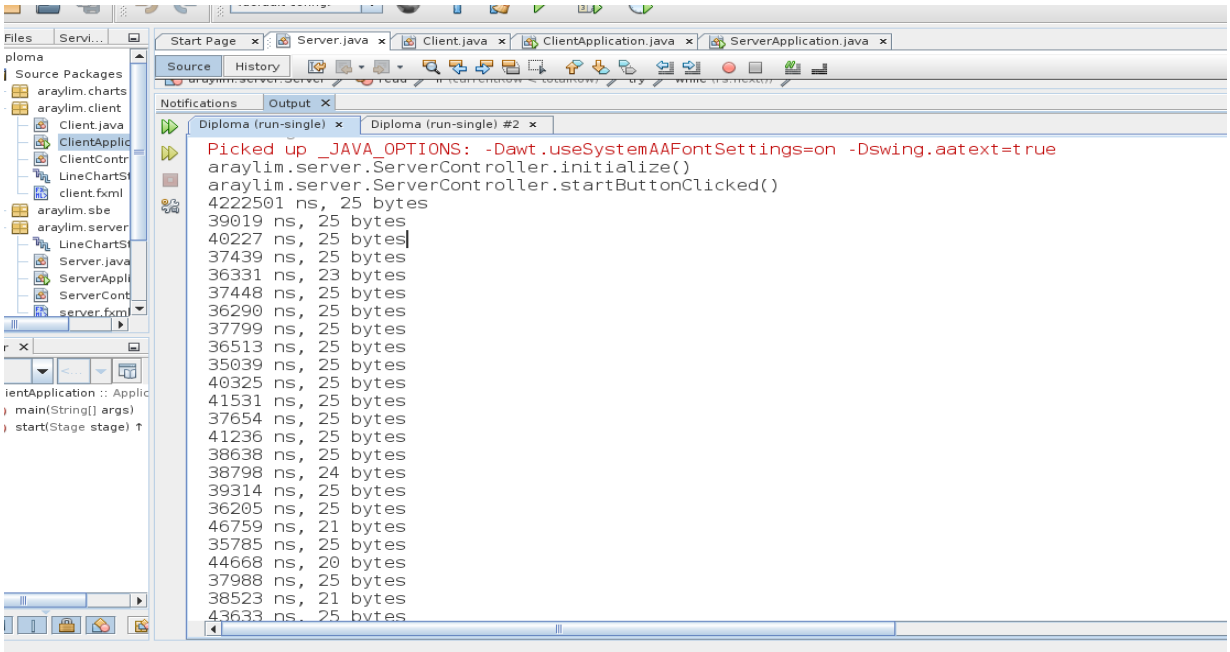
Приложение А

Метод run()

```
@Override
public void run() {
    if(DEBUG)System.out.println("araylim.server.Server.run()");
    try{
        String sql;
        sql = "SELECT COUNT(*) AS total FROM EUR_USD_M1";
        ResultSet rs = statement.executeQuery(sql);
        while(rs.next()) totalRow = rs.getInt("total");

        while(!Thread.currentThread().isInterrupted()){
            selector.select(2000);

            Iterator<SelectionKey> keys = selector.selectedKeys().iterator();
            while (keys.hasNext()) {
                SelectionKey key = keys.next();
                keys.remove();

                if(!key.isValid()){
                    continue;
                }

                if (key.isAcceptable()) {
                    accept(key);
                } else if (key.isWritable()) {
                    write(key);
                } else if (key.isReadable(){
                    read(key);
                }

            -
```

Приложение Б

Результаты размера сообщения SBE



Приложение В

Результаты размера сообщения JSON