Инженерный вестник Дона, №3 (2025)
ivdon.ru/ru/magazine/archive/n3y2025/9884

Обзор решений для оптимизации системы управления комплексом защиты объекта

В.Р. Скрипова, К.А. Аксенов

Уральский федеральный университет

Аннотация: Оптимизация автоматизированных систем управления комплексами защиты объектов не теряет своей актуальности. В данной статье представлен обзор инструментов для организации отдельных процессов мониторинга: опроса устройств, обработки полученных данных и передачу данных в пользовательский интерфейс. На основе анализа рассмотренной информации планируется сформировать базу решений для системы управления комплексом технических средств. В ходе исследования удалось выявить, что многопоточная архитектура сетевого менеджера в комбинации с адаптивным алгоритмом позволяет осуществить крупномасштабный опрос, алгоритмы кластеризации и настройка платформ обработки большого объема данных повышают производительность, а протокол WebSocket эффективен при передаче данных. Результатом оценки стал набор средств для разработки программно-аппаратного комплекса.
Ключевые слова: датчик, система управления, мониторинг, менеджер SNMP, кластеризация, Hadoop, MapReduce, Spark, Apache Kafka, WebSocket.

Введение

На сегодняшний день область защиты особо важных объектов от внешних угроз является перспективным направлением для разработки новых оптимальных комплексных решений. Непрерывное развитие в области аппаратного и программного обеспечения требует усовершенствования разработанных ранее стабильных систем. Главными составляющими комплекса защиты объекта являются совокупность технических средств и автоматизированная распределенная информационная система управления. Одной из главных функциональных задач системы является организация процесса мониторинга технических средств (отслеживание текущего состояния), который включает в себя три основные стадии: опрос устройств (сбор данных), обработка полученных данных, передача данных в пользовательский интерфейс в режиме реального времени (рис. 1). В данной статье рассматриваются технические и программные решения, методы и алгоритмы оптимизации процессов опроса (сбора), обработки и передачи

данных. Цель исследования – выявить наиболее эффективные решения. Критериями оценки выступают требования к масштабируемости, производительности, отказоустойчивости и быстродействию. Основными методами исследования являются анализ и сравнение. Соответственно, можно выделить ряд задач исследования: сделать краткий обзор решений, дать оценку эффективности решений, подвести итоги результатов оценки.

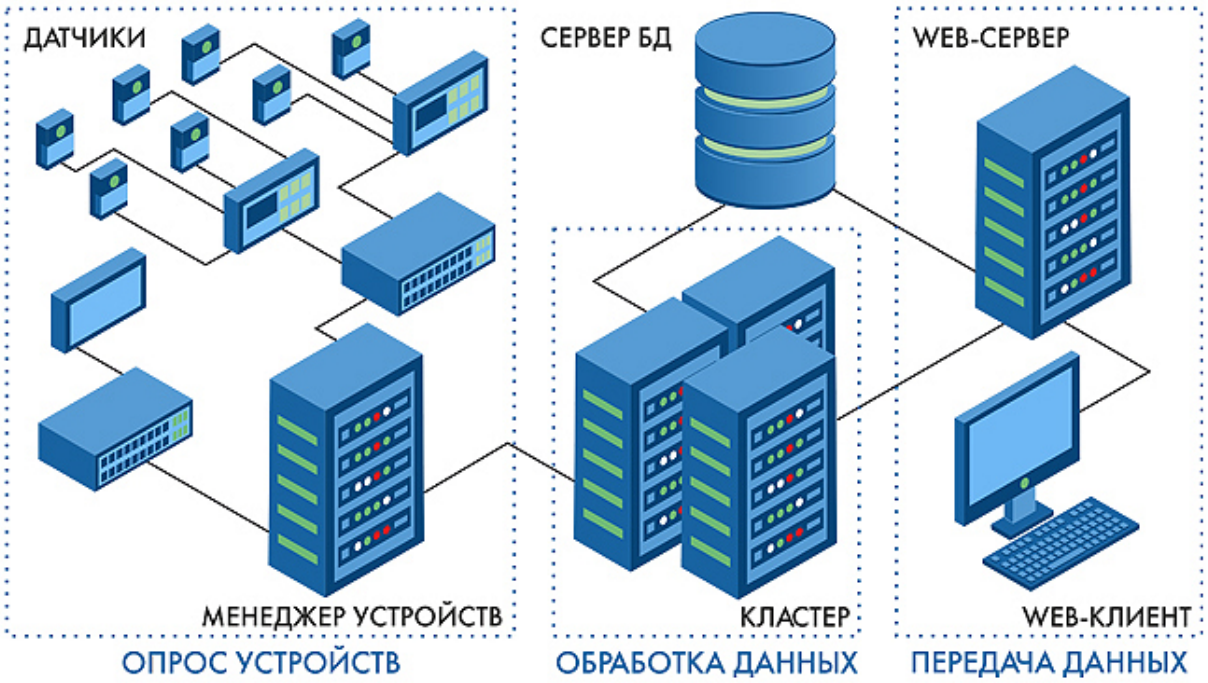

Рис. 1. – Архитектура системы программно-аппаратного комплекса. Иллюстрация автора.

Обзор решений

1. Опрос устройств (сбор данных)

Популярным протоколом мониторинга является SNMP (Simple Network Management Protocol). Информация управления хранится в базах управляющей информации MIB (Management Information Base), элементы которых распознаются идентификаторами объектов OID (Object Identifier). Менеджер SNMP опрашивает агентов на контролируемых устройствах с помощью OID и получает нужные значения [1]. В качестве базового протокола используется UDP (User Datagram Protocol).

Адаптация гетерогенных данных

Для крупномасштабного опроса используется многопоточная архитектура менеджера. Адаптация к любому типу MIB выполняется за счет того, что агент SNMP представляется в виде объекта SnmpAgent, содержащий адрес и OID. Для выполнения запросов создаётся массив объектов SnmpAgent. Затем используется пул потоков с фиксированным количеством потоков, где каждый поток извлекает агентов из массива и выполняет запросы [1].

Адаптивный алгоритм опроса

Суть алгоритма заключается в настройке нескольких сегментов временных интервалов опроса, каждый из которых имеет свой собственный пул потоков. Изначально, всем агентам назначается сегмент с наименьшим временем отклика. В зависимости от времени отклика агентов в рамках отдельного интервала опроса происходит повышение или понижение их рейтинга [1] (рис. 2), т.е. перемещение агента в сегмент с более высокой или низкой частотой опроса.

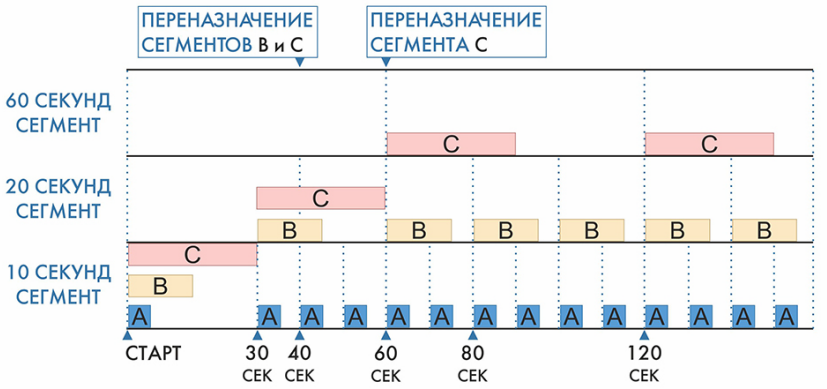

Рис. 2. – Понижение рейтинга агентов B и C в рамках интервала опроса [1].

Стабилизация процесса перемещения выполняется за счет использования планировщика (наличие у каждого агента фиксированного или динамического счетчика количества повторяющегося времени отклика).

2. *Обработка данных*

Алгоритм управляемой кластеризации и периферийных вычислений

Кластеризация – это типизация похожих объектов на разные группы по какому-либо признаку – разбиение набора входных данных на некоторые подмножества (кластеры), так, что данные в каждом подмножестве имеют какую-либо общую характеристику [2]. Разбиение выборки на группы позволяет упростить дальнейшую обработку данных и принимать решения по каждому кластеру [3].

Кластеризация потоковых данных в реальном времени выполняется с помощью двух процессов. Сначала кортежи данных, обрабатываются путем грубой кластеризации, определяется количество макрокластеров и положение центральной точки, формируется набор отличных друг от друга макрокластеров. На втором этапе реализуется точная кластеризация, производится выборка набора макрокластеров и выполняется параллельная кластеризация методом k-средних с наибольшим и наименьшим расстояниями. Затем, алгоритм кластеризации и алгоритм периферийных вычислений объединяются для кластеризации в рамках структуры периферийных вычислений [4]. Вычислительная задача выполняется на вычислительных ресурсах, близких к источнику данных. Запрос и ответ в алгоритме являются двусторонними. Терминальное устройство не только передает запрос в облачный вычислительный центр, но и выполняет вычислительные задачи. Некоторые сервисы могут предоставлять ответы непосредственно на периферии и возвращаться к терминальному устройству, облачному вычислительному центру и периферийным ресурсам [4]. Таким образом, формируются два отдельных потока ответов на запросы.

Параллельный генетический алгоритм в Apache Hadoop MapReduce

Apache Hadoop MapReduce – это среда, предназначенная для распределенной и параллельной обработки больших наборов данных, является компонентом обработки данных открытой платформы Apache Hadoop. Суть парадигмы MapReduce заключается в разделении большого

набора данных на части и в параллельной обработке на нескольких компьютерах кластера. Фреймворк MapReduce включает компоненты: Map и Reduce. Компонент Map принимает входные данные и преобразует их в набор промежуточных пар ключ-значение. Компонент Reduce принимает выходные данные компонента Map, обрабатывает их и выдает окончательный результат [5] (рис. 3).

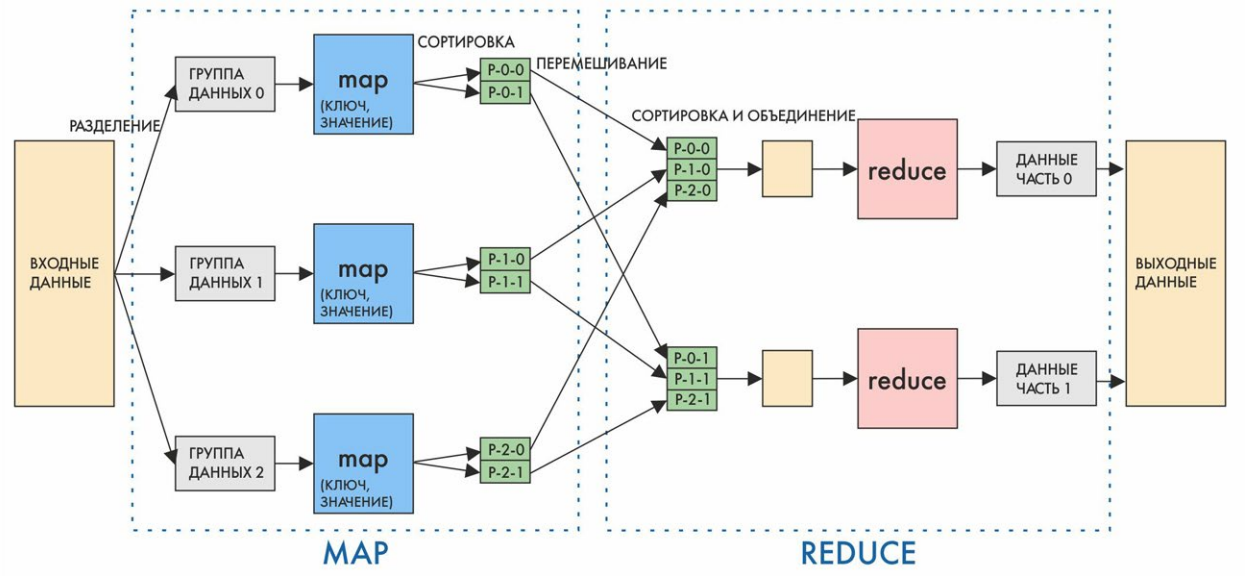

Рис. 3. – Архитектура Apache Hadoop MapReduce [6].

Параллельные генетические алгоритмы являются разновидностью генетического алгоритма, использующего несколько процессоров для ускорения процесса поиска. Суть алгоритма заключается в двухфазовой кластеризации. Первая фаза (стадия Map) содержит следующие этапы: 1) Инициализация популяции (центроиды групп выбираются случайным образом из полученных разделенных данных); 2) Оценка приспособленности (вычисляется Индекс Девида-Булдина для каждой особи); 3) Мутация и селекция (Скрещивание: потомок центроид которого среднеарифметическое центроида родителей; Мутационный обмен: используется до 9 дополнительных точек данных; Селекция: отбор по турниру); 4) Обновление (новая популяция заменяет старую). Вся процедура продолжается до выполнения условия завершения (максимальное число итераций). Затем

самые адаптированные особи конечной популяции передаются на вторую фазу (стадия Reduce), где производится объединение и выбор наилучшей особи, т.е. оптимального решения задачи (использование метода превышения порогового значения центроидов).

Apache Spark

Apache Spark – это распределенная вычислительная платформа в памяти, основанная на парадигме MapReduce, предназначенная для обработки большого объема данных. Параметры конфигурации Spark контролируют параллелизм, вычислительные ресурсы, сжатие и операции ввода-вывода. Архитектура Spark представляет собой схему, где узел драйвера является мастером, а рабочие узлы – подчиненными. Узел драйвера распределяет задачи по рабочим узлам кластера. Каждый узел имеет исполнитель, который выполняет назначенные ему задачи. Устойчивый распределенный набор данных (УРНД) распределяется по узлам кластера путем применения к данным операций трансформации и действия. Трансформации применяют функцию к существующему УРНД, в результате чего создается новый УРНД. Действия возвращают значение в программу драйвера или сохраняют его в системе хранения. Каждый УРНД хранит историю примененных трансформаций. В случае потери данных Spark достигает отказоустойчивости, используя историю для восстановления потерянных данных [7].

Apache Kafka

Apache Kafka предоставляет сервис потоковой передачи данных, используя систему обмена сообщениями «публикация-подписка». Топик – это элемент Kafka, который объединяет все сообщения, относящиеся к одной тематике. Платформа Apache Kafka использует разделы для масштабирования топика между множеством брокеров, чтобы производители могли записывать, а потребители считывать данные параллельно. Каждый

раздел может быть реплицирован на брокерах для обеспечения отказоустойчивости. Реплики помогают поддерживать доступность в случае, если один из брокеров выходит из строя. Запросы как от производителя, так и от потребителя к разделу обслуживаются через лидирующую реплику [8]. Модель конфигурации Apache Kafka состоит из: *П* – производителей (отправляют данные), *Т* – топик (логическая категория данных), хранящийся в наборе из *Б* брокеров (кластер), *Р* – разделы брокеров, *р* – реплика (копия), *п* – потребителей (читают данные из топика) (рис. 4).

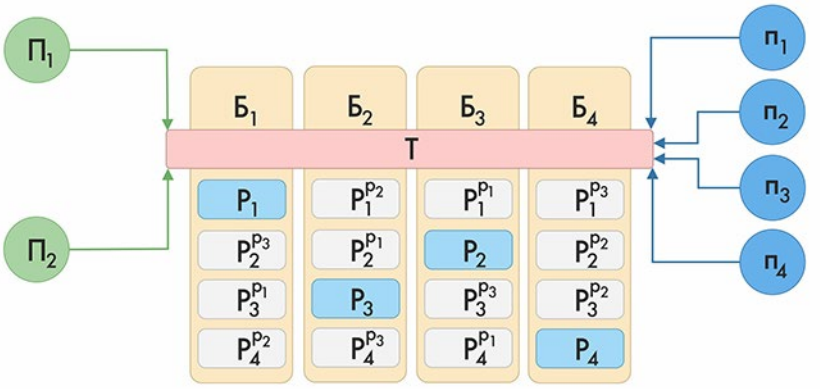

Рис. 4. – Пример конфигурации Apache Kafka [8].

3. Протоколы передачи

HTTP polling (Hypertext Transfer Protocol)

Стремление к режиму реального времени в HTTP polling осуществляется за счет отправки частых HTTP-запросов на сервер, который отправляет пустой или базовый ответ (рис. 5). Когда данные становятся доступными, клиент получает ответ от сервера с задержкой примерно равной времени между запросами. Опция «заголовок keep-alive» позволяет повторно использовать TCP-соединение (Transmission Control Protocol) для нескольких запросов, что снижает ресурсы на создание нового соединения для каждого запроса [9].

HTTP long polling

Протокол HTTP long polling – это оптимизированный вариант HTTP polling, при котором сервер удерживает запросы клиентов открытыми до тех

пор, пока не появится новое сообщение для отправки клиенту или не истечет время ожидания [10] (рис. 5), что снижает количество бесполезных запросов к серверу и значительно уменьшает объем передаваемых байтов.

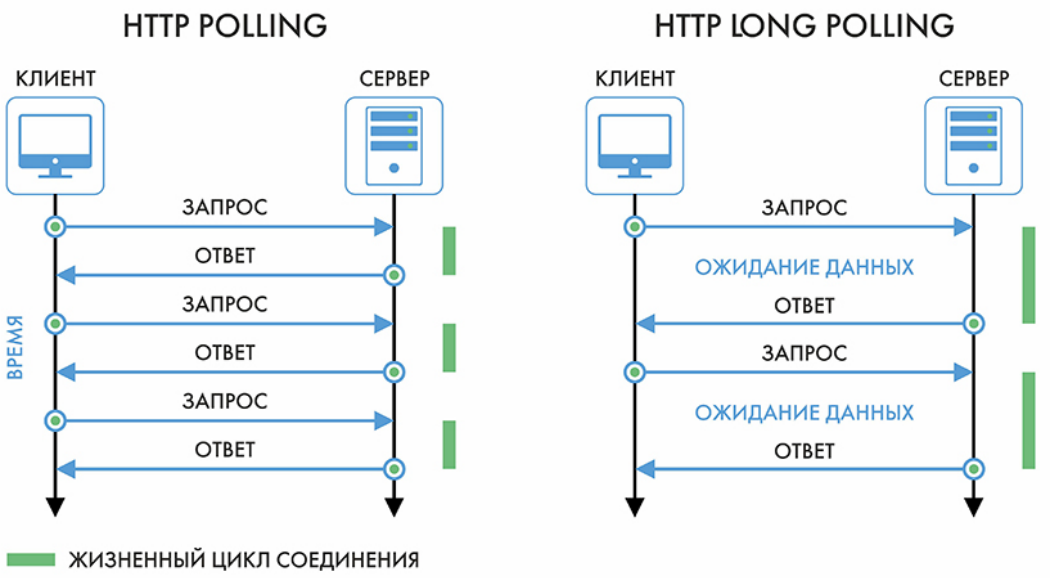

Рис. 5. – Схема работы Http polling и Http long polling. Иллюстрация автора.

Server-Sent Events (HTTP streaming)

Протокол Server-Sent Events (SSE) – это вариант протокола HTTP streaming, поддерживающий открытым базовое TCP-соединение даже после доставки данных в ответ на запрос, что способствует передачу поступающих от сервера доступных дополнительных данных (рис. 6). Достигается это с помощью заголовка Transfer Encoding: chunked HTTP, что позволяет удерживать TCP-соединение открытым и устраняет необходимость в повторном подключении каждый раз, когда сервер отправляет данные [9]. Реализация SSE осуществляется интерфейсом JavaScript EventSource предоставляемый API браузера (Application Programming Interface).

WebSocket

Протокол WebSocket обеспечивает полнодуплексный двунаправленный канал связи в рамках одного сокета [11]. Старт работы протокола осуществляется handshake (рукопожатием). Клиент инициирует соединение, отправляя HTTP GET запрос с заголовком Upgrade: websocket. Если сервер

способен обслуживать соединение WebSocket, он отвечает HTTP 101: Switching Protocols, и соединение устанавливается. После этого канал связи с полудуплексным режимом доступен как для клиента, так и для сервера. Клиент может получать обновления от сервера (текстовые, бинарные данные) в реальном времени, без необходимости опрашивать сервер (рис. 6). Все основные (настольные и мобильные) браузеры предоставляют полную поддержку WebSocket. WebSocket прикрепляет небольшой заголовок (2 – 14 байт), который содержит код операции, размер нагрузки и бит маскировки. Браузер автоматически закрывает соединения, когда клиент закрывает или переходит со страницы [9].

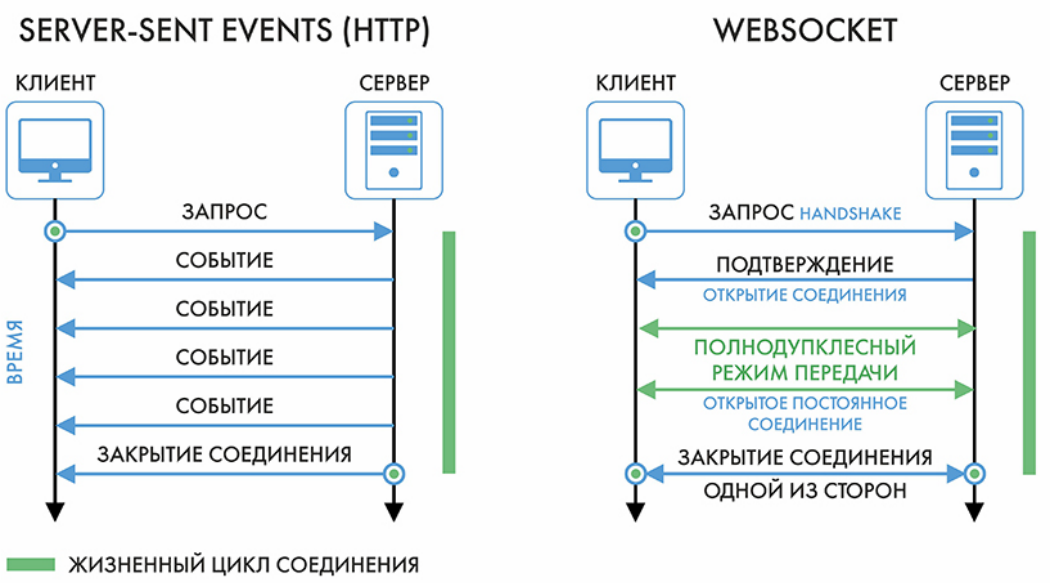

Рис. 6. – Схема работы Server-Sent Events и WebSocket. Иллюстрация автора.

Оценка эффективности решений

Опрос устройств (сбор данных). Оценка эффективности

Однопоточная архитектура SNMP-менеджера проста в разработке, но применима только с небольшим количеством агентов. Масштабирование экземпляров менеджера требует аппаратных и программных ресурсов, а также снижают производительность. Напротив, гибкая многопоточная архитектура эффективно сокращает аппаратные ресурсы для опроса

растущего числа агентов. А компонент сценария для быстрой адаптации к гетерогенным данным позволяет избежать частой настройки менеджера или использование уровня трансляции [12]. Применение адаптивного алгоритма опроса позволяет не перегружать агент несколькими параллельными запросами. Если поток занят медленным агентом, остальная часть пула потоков продолжает опрашивать другие агенты, тем самым сокращая время выполнения всех запросов. Согласно результатам экспериментов источника информации [1], при отсутствии планировщика, 1,36% всех запросов выполнялись дольше и наблюдалась нагрузка на ЦП (центральный процессор) агента. При наличии одного из планировщиков, в 0,24% случаев агенты быстрее перемещались между сегментами, что предотвращало увеличение нагрузки на ЦП. Выбор фиксированного или динамического планировщика обеспечивает баланс между защитой от нагрузки на ЦП агента и более высокой частотой опроса.

Обработка данных

Оценка эффективности алгоритма управляемой кластеризации и периферийных вычислений. Результаты экспериментов, описанных в статье [4], показали, что кластеризация потоковых данных в реальном времени с помощью алгоритма, снижает уровень ошибочной классификации (подтверждается результатами сравнения полученного решения в ходе кластеризации предложенного алгоритма и алгоритма k-средних), обладает хорошей стабильностью (поддерживает заданное количество кластеров), имеет лучшую производительность, позволяет быстро получить глобальное оптимальное решение (поддерживает фиксированное значение времени работы алгоритма) и обрабатывает большие объемы данных с высокой скоростью в режиме реального времени (подход двойной кластеризации).

Оценка эффективности параллельного генетического алгоритма. В рассмотренном документе [5] авторы настроили Hadoop MapReduce,

реализовав двухфазную кластеризацию, что представляет собой новый метод распараллеливания кластеризации на основе генетического алгоритма. В ходе эксперимента по оценке производительности параллельного генетического алгоритма и последовательного генетического алгоритма было выполнено 500 итераций. Тестирование было реализовано на кластере, состоящем из 5 узлов, оснащенных Hadoop v1.2.1, 2 ГБ оперативной памяти, жестким диском объемом 250 ГБ и по 2 процессорных ядра. Набор данных представлял собой дифференциальные координаты карты Европы, состоявший из 169308 экземпляров и 2 измерений. При параллельном генетическом алгоритме было зафиксировано ускорение на 80% с точностью кластеризации 92%. Данный факт подтверждает, что использование генетических алгоритмов повышает точность и эффективность процесса кластеризации.

Сравнительный анализ Hadoop и Spark. На основе результатов исследования, описанного в литературе [13], можно сделать вывод, что производительность систем Hadoop и Spark во многом зависит от размера входных данных, а также от правильного выбора и настройки параметров. Для исследования производительности выполнения рабочих нагрузок WordCount и TeraSort (из пакета HiBench) при настройке 18 различных значений параметров, был проведен ряд экспериментов на реализованном в лаборатории кластере, состоящим из девяти узлов с объемом набора данных 600 ГБ. Авторы обнаружили, что производительность Spark в 2 раза выше Hadoop при рабочей нагрузке WordCount и в 14 раз при рабочей нагрузке TeraSort. Результаты по пропускной способности и ускорению показали, что Spark более стабилен и быстрее, благодаря способности обрабатывать данные в памяти.

Оценка эффективности Apache Kafka. Apache Kafka стал одним из самых успешных продуктов с открытым исходным кодом от Apache и

получил широкое признание в сообществе распределенных систем. Вопрос об оптимальной настройки эффективного распределения топика по разделам для повышения производительности кластера Apache Kafka, все еще представляет собой актуальную исследовательскую задачу. Авторы статьи [8] сформулировали задачу поиска оптимального разделения топика и показали ее вычислительную неразрешимость. Кроме того, они разработали два эвристических алгоритма для решения проблемы (первый пытается минимизировать, а второй – максимизировать количество брокеров, используемых в кластере) и провели оценку производительности, опираясь на рекомендованную Microsoft и Confluent настройку конфигурации разделения топиков в Apache Kafka. Было продемонстрировано, что оба эвристических алгоритма используют системные ресурсы более эффективно, чем рекомендованные, и хорошо учитывают ограничения, связанные с задержкой репликации, использованием ресурсов, загрузкой операционной системы и режима недоступности.

Передача данных. Сравнительный анализ

Согласно обзору протоколов передачи, HTTP вариации протокола не могут обеспечить полнодуплексную связь, имеют ограничения в типе передаваемых данных (только текстовые), содержат бесполезные запросы (пустой ответ) или вызывают задержки ответа от сервера. Решения страдают от расходов ресурсов на таймауты соединения и повторения заголовков HTTP сообщений. Весь процесс увеличивает нагрузку на сервер и сеть. Протокол WebSocket обеспечивает значительное снижение сетевых издержек, эффективную транспортировку данных в реальном времени с малой задержкой и высокой пропускной способностью, а также предоставляет простой API, который поддерживается во всех современных браузерах [11].

Результаты исследования

На основании изученного материала можно сделать вывод о том, что реализация процессов сбора, обработки, передачи данных отвечающих требованиями масштабируемости, производительности, отказоустойчивости и быстродействию зависит от следующих факторов: разработки многопоточной архитектуры и применения адаптивного алгоритма для сетевых менеджеров; выбора оптимального алгоритма кластеризации данных; эффективная настройка параметров платформ, обеспечивающих обработку больших данных в рамках кластера; использование протокола, поддерживающего полнодуплексный двунаправленный канал связи между клиентом и сервером.

Для оптимизации системы управления комплексом защиты объектов приемлемы решения для сетевого менеджера, платформа Apache Kafka, поддерживающая обработку потока данных из различных индивидуальных источников, и протокол WebSocket, передающий данные в пользовательский интерфейс в режиме реального времени.

Заключение

В данной статье были рассмотрены решения по разработке архитектуры сетевого менеджера, алгоритмы кластеризации, функциональные возможности платформ для обработки данных и характеристики протоколов передачи данных. Результатом оценки стала база решений для архитектуры системы управления комплексом защиты объекта. В дальнейшем, планируется провести исследования в области оптимизации и реализации каждого отдельного звена архитектуры системы. Поскольку рассмотренные в статье процессы мониторинга применяются в различных сферах, начиная от промышленности и заканчивая экологией, то данный обзор также будет иметь интерес для исследователей в других областях.

Литература

1. Roquero Paula, Aracil Javier. On Performance and Scalability of Cost-Effective SNMP Managers for Large-Scale Polling // IEEE Access, 2021, Volume 9, pp. 7374 – 7383. URL: ieeexplore.ieee.org/document/9314056.

2. Чернов А.В., Бутакова М.А., Шевчук П.С. Кластеризация данных методом растущего нейронного газа // Инженерный вестник Дона, 2020, №7. URL: ivdon.ru/ru/magazine/archive/n7y2020/6537.

3. Голубева А.О., Виноградова Г.Л. Кластеризация процессов промышленного предприятия в методе их адаптации под заказ // Инженерный вестник Дона, 2012, №2. URL: ivdon.ru/ru/magazine/archive/n2y2012/829.

4. Li Xiang, Zhang Zijia. Research and Analysis for Real-Time Streaming Big Data Based on Controllable Clustering and Edge Computing Algorithm // IEEE Access, 2019, Volume 7, pp. 171621 – 171632. URL: ieeexplore.ieee.org/abstract/document/8913543.

5. Gautam Chandra Shekhar, Soni Laxmi Narayan, Pandey Prabhat. Clustering of Bigdata Using Genetic Algorithm in Hadoop MapReduce // Conference: International Conference on Computer Science and Engineering. India, Lonavala, 2023. URL: researchgate.net/publication/374355962.

6. Veiga Jorge, Expósito Roberto R., Taboada Guillermo L., Touriño Juan. Analysis and evaluation of MapReduce solutions on an HPC cluster // Computers & Electrical Engineering, 2015. URL: researchgate.net/publication/288686907.

7. Alfailakawi Mohammad Gh., Aljame Maryam, Ahmad Imtiaz. Parallel and Distributed Implementation of Sine Cosine Algorithm on Apache Spark Platform // IEEE Access, 2021, Volume 9, pp. 77178 – 77202. URL: ieeexplore.ieee.org/document/9435347.

8. Raptis Theofanis P., Passarella Andrea. On Efficiently Partitioning a Topic in Apache Kafka // Conference: International Conference on Computer,

Information and Telecommunication Systems. Greece, Piraeus, 2022. URL: ieeexplore.ieee.org/document/9832981.

9. Murley Paul, Ma Zane, Mason Joshua, Bailey Michael, Kharraz Amin. WebSocket Adoption and the Landscape of the Real-Time Web // Conference: Web Slovenia, Ljubljana, 2021. URL: researchgate.net/publication/352113552.

10. Naik Aniket Avinash, Khare Meghana R. Study of WebSocket Protocol for Real-Time Data Transfer // International Research Journal of Engineering and Technology, 2020, Volume 7. URL: academia.edu/44285187.

11. Ogundeyi K. E., Yinka-Banjo C. WebSocket in real time application // Nigerian Journal of Technology, 2019, Volume 38, Number 4, pp. 1010 – 1020. URL: ajol.info/index.php/njt/article/view/191780.

12. Chavan Santosh S., Madanagopal R. Generic SNMP Proxy Agent Framework for Management of Heterogeneous Network Elements // Conference: First International Communication Systems and Networks and Workshops. India, Bangalore, 2009. URL: ieeexplore.ieee.org/document/4808873.

13. Ahmed N., Barczak Andre L. C., Susnjak Teo, Rashid Mohammed A. A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench // Journal of Big Data, 2020, Volume 7. URL: researchgate.net/publication/347137855.

References

1. Roquero Paula, Aracil Javier. IEEE Access, 2021, Volume 9, pp. 7374 – 7383. URL: ieeexplore.ieee.org/document/9314056.

2. Chernov A.V., Butakova M.A., Shevchuk P.S. Inzhenernyj vestnik Dona, 2020, №7. URL: ivdon.ru/ru/magazine/archive/n7y2020/6537.

3. Golubeva A.O., Vinogradova G.L. Inzhenernyj vestnik Dona, 2012, №2. URL: ivdon.ru/ru/magazine/archive/n2y2012/829.

4. Li Xiang, Zhang Zijia. IEEE Access, 2019, Volume 7, pp. 171621 – 171632. URL: ieeexplore.ieee.org/abstract/document/8913543.

5. Gautam Chandra Shekhar, Soni Laxmi Narayan, Pandey Prabhat. Conference: International Conference on Computer Science and Engineering. India, Lonavala, 2023. URL: researchgate.net/publication/374355962.

6. Veiga Jorge, Expósito Roberto R., Taboada Guillermo L., Touriño Juan. Computers & Electrical Engineering, 2015. URL: researchgate.net/publication/288686907.

7. Alfailakawi Mohammad Gh., Aljame Maryam, Ahmad Imtiaz. IEEE Access, 2021, Volume 9, pp. 77178 – 77202. URL: ieeexplore.ieee.org/document/9435347.

8. Raptis Theofanis P., Passarella Andrea. Conference: International Conference on Computer, Information and Telecommunication Systems. Greece, Piraeus, 2022. URL: ieeexplore.ieee.org/document/9832981.

9. Murley Paul, Ma Zane, Mason Joshua, Bailey Michael, Kharraz Amin. Conference: Web Slovenia, Ljubljana, 2021. URL: researchgate.net/publication/352113552.

10. Naik Aniket Avinash, Khare Meghana R. International Research Journal of Engineering and Technology, 2020, Volume 7. URL: academia.edu/44285187.

11. Ogundeyi K. E., Yinka-Banjo C. Nigerian Journal of Technology, 2019, Volume 38, Number 4, pp. 1010 – 1020. URL: ajol.info/index.php/njt/article/view/191780.

12. Chavan Santosh S., Madanagopal R. Conference: First International Communication Systems and Networks and Workshops. India, Bangalore, 2009. URL: ieeexplore.ieee.org/document/4808873.

13. Ahmed N., Barczak Andre L. C., Susnjak Teo, Rashid Mohammed A. Journal of Big Data, 2020, Volume 7. URL: researchgate.net/publication/347137855.

Дата поступления: 30.12.2024

Дата публикации: 25.02.2025