УДК 005.591.1

МЕТОДЫ ОПТИМИЗАЦИИ АГРЕГАЦИИ СОБЫТИЙ В СИСТЕМАХ REAL-TIME BIDDING ПРИ ВЫСОКИХ НАГРУЗКАХ

Анциферова Е.И., магистрант гр.467041

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

Давыдова Н.С. – канд. тех. наук, доцент

Аннотация. В работе представлен аналитический обзор методов оптимизации агрегации событий в системах Real-Time Bidding в условиях высоких нагрузок. Рассматриваются структурные особенности RTB-логов и ключевые проблемы их обработки, обусловленные высокой интенсивностью поступления и значительными объёмами данных. Систематизированы современные подходы к агрегации, включая потоковую обработку, методы предварительной агрегации и применение колоночных хранилищ данных. Проведён сравнительный анализ преимуществ и ограничений рассматриваемых решений. Полученные результаты могут быть использованы при проектировании масштабируемых систем обработки данных в рекламных платформах.

Ключевые слова. Real-Time Bidding, RTB-логи, агрегация событий, потоковая обработка данных, предварительная агрегация, колоночные базы данных, ClickHouse, Apache Kafka, высоконагруженные системы, масштабируемость, аналитика данных.

Современный рынок цифровой рекламы характеризуется стремительным ростом объёмов обрабатываемых данных и повышением требований к скорости принятия решений. Системы Real-Time Bidding (RTB), обеспечивающие программную закупку рекламных мест в режиме реального времени, генерируют десятки миллионов событий в секунду, что создаёт значительные вызовы для инфраструктуры обработки и анализа данных. Согласно отчётам аналитических агентств, глобальный объём транзакций в RTB-экосистеме превысил 500 миллиардов запросов в сутки, при этом средняя задержка на обработку одного запроса не должна превышать 100 миллисекунд [1].

В условиях таких нагрузок традиционные методы агрегации и аналитики данных оказываются неэффективными, что обусловливает необходимость разработки и применения специализированных подходов к оптимизации обработки событий.

Ключевой задачей RTB-платформ является агрегация разнородных событий – показов (impressions), кликов (clicks), конверсий (conversions) и биддинговых запросов – с целью формирования аналитических метрик в режиме, близком к реальному времени.

При этом система должна обеспечивать не только высокую пропускную способность, но и точность расчётов, масштабируемость и отказоустойчивость. Актуальность исследования определяется следующими факторами:

– Экспоненциальный рост объёмов данных. Количество рекламных запросов увеличивается на 40–50% ежегодно, что требует постоянной модернизации систем обработки.

– Жёсткие требования к задержкам. Рекламодатели и аналитики нуждаются в оперативном доступе к метрикам эффективности кампаний для принятия управленческих решений.

– Сложность архитектур. Гетерогенность источников данных, распределённая природа систем и необходимость обеспечения консистентности данных усложняют проектирование решений.

– Экономическая эффективность. Оптимизация процессов агрегации напрямую влияет на инфраструктурные затраты платформ.

Real-Time Bidding представляет собой автоматизированную систему программной закупки рекламы, функционирующую на основе аукционных механизмов в режиме реального времени. Типичная RTB-экосистема включает ряд ключевых компонентов [2], среди которых SSP (платформа на стороне издателя), DSP (платформа на стороне рекламодателя), рекламная биржа (Ad Exchange), координирующая процесс торгов, а также DMP, обеспечивающая управление данными для таргетинга.

Процесс обработки рекламного запроса включает последовательность этапов, начиная с формирования запроса на участие в аукционе (bid request), содержащего информацию о пользователе и контексте. Далее DSP-платформы формируют и отправляют ставки, после чего проводится аукцион, чаще всего по модели second-price auction, определяется победитель и осуществляется показ рекламного объявления. В случае взаимодействия пользователя фиксируются события клика и конверсии. Существенным ограничением является требование к времени обработки: полный цикл должен укладываться в 100–120 миллисекунд.

RTB-системы генерируют различные типы событий, включая запросы на участие в аукционе, ставки, показы, клики и конверсии [1]. Эти события отличаются как по частоте возникновения, так и по объему и структуре данных. Наиболее массовыми являются bid requests и ставки, достигающие миллионов событий в секунду, тогда как клики и конверсии происходят значительно реже, но обладают высокой ценностью с точки зрения аналитики. Каждое событие содержит набор атрибутов,

таких как идентификаторы пользователей, параметры рекламных кампаний, временные метки и контекст размещения.

Данные, формируемые в RTB-системах, характеризуются значительными объемами, высокой скоростью поступления и разнообразием форматов. Объем логов может достигать десятков или сотен терабайт в сутки, а в крупных системах – вплоть до петабайтных масштабов. Поток событий отличается неравномерностью нагрузки, обусловленной временными факторами, а сами данные имеют множество версий схем. Дополнительную сложность представляет временная специфика обработки: события могут поступать с задержкой, нарушать хронологический порядок и требовать учета как времени генерации, так и времени обработки. Кроме того, данные между различными типами событий связаны, что требует выполнения операций объединения и построения пользовательских сессий.

Указанные особенности определяют высокие требования к системам агрегации данных. Такие системы должны обеспечивать обработку миллионов событий в секунду при минимальной задержке формирования аналитических показателей. Важным аспектом является точность вычислений в условиях неполных и запаздывающих данных, а также способность системы масштабироваться с ростом нагрузки. Не менее значимыми являются требования отказоустойчивости и поддержки гибких аналитических запросов. Совокупность этих факторов обусловливает необходимость применения специализированных методов оптимизации агрегации, рассматриваемых в дальнейшем.

Одной из центральных проблем агрегации RTB-событий является обеспечение минимальных задержек между моментом возникновения события и его отражением в агрегированных метриках. Анализ показывает, что задержка складывается из нескольких компонентов:

$$T_{total} = T_{network} + T_{ingestion} + T_{processing} + T_{storage} + T_{query},$$

где $T_{network}$ – задержка передачи данных по сети (10–50 мс), $T_{ingestion}$ – время приёма данных в систему обработки (20–100 мс), $T_{processing}$ – время вычисления агрегатов (100 мс – 10 сек), $T_{storage}$ – запись результатов (50–500 мс), T_{query} – выполнение аналитических запросов (100 мс – 5 сек).

Высокая интенсивность записи событий создает существенную нагрузку на системы хранения данных. При выполнении агрегации непосредственно на уровне базы данных возникает эффект усиления записи (write amplification), при котором обработка каждого события приводит к множественным операциям обновления. Дополнительно наблюдается конкуренция за блокировки (lock contention), обусловленная параллельным обновлением агрегатов, что снижает общую производительность системы. Существенное влияние оказывает и необходимость поддержания индексов для выполнения многомерных аналитических запросов, что увеличивает стоимость операций записи в несколько раз.

Вертикальное масштабирование, предполагающее увеличение мощности отдельных серверов, ограничено физическими возможностями оборудования и сопровождается нелинейным ростом стоимости. Горизонтальное масштабирование через шардинг или партиционирование сталкивается с рядом проблем: выполнение кросс-партиционных запросов требует координации между узлами, данные распределяются неравномерно (data skew), когда отдельные кампании генерируют значительно больше событий, а перераспределение данных при добавлении новых узлов создает дополнительные накладные расходы. Кроме того, точность агрегатов усложняется из-за поступления событий вне хронологического порядка (out-of-order events), что требует ретроактивных корректировок и создает компромисс между полнотой данных и актуальностью метрик.

Потоковая обработка данных основана на концепции непрерывной обработки событий по мере их поступления, в противовес традиционной пакетной обработке накопленных данных. Формально, поток событий можно представить как бесконечную последовательность:

$$S = \{e_1, e_2, e_3, \ldots, e_n, \ldots\},$$

где каждое событие $e_i = (\text{key}, \text{value}, \text{timestamp})$.

Агрегация в потоковой модели выполняется инкрементально:

$$A_t = f(A_t - 1, e_t),$$

где A_t – состояние агрегата в момент t, f – функция обновления.

Далее представлены современные технологические платформы для потоковой обработки данных, широко применяемые в системах Real-Time Bidding, с анализом их возможностей по обеспечению высокой производительности, низкой задержки и масштабируемости.

Apache Kafka Streams представляет собой клиентскую библиотеку для создания распределённых потоковых приложений. Она поддерживает stateful processing через RocksDB, гарантирует идемпотентность операций при сбоях и позволяет строить обработку данных в виде

топологий «источник → процессор → приёмник». Производительность при простой агрегации может достигать до 10^6 событий в секунду на узел.

Apache Flink обеспечивает низколатентную обработку с поддержкой семантики event-time, включая механизмы Watermarks для отслеживания времени событий, гибкое управление состояниями через различные backend'ы и асинхронное создание снимков состояния для восстановления. Flink эффективно обрабатывает события вне хронологического порядка с задержкой менее 100 мс.

Apache Spark Streaming использует микробатчинговый подход и абстракцию непрерывной таблицы. Задержка обработки варьируется от 100 мс в режиме continuous до нескольких секунд в micro-batch режиме, при этом обеспечивается высокая пропускная способность потоков данных.

Сравнительный анализ технологических платформ расположен в таблице 1.

Таблица 1 – Сравнение параметров технологических платформ

Платформа	Задержка	Пропускная способность	Управление состоянием	Сложность
Kafka Streams	10–100 мс	10^6/сек/узел	RocksDB	Низкая
Flink	5–50 мс	10^6/сек/узел	Flexible	Средняя
Spark Streaming	100 мс – 5 сек	10^7/сек/кластер	In-memory/RDD	Средняя

Потоковая обработка обеспечивает непрерывное вычисление агрегатов по мере поступления событий, что позволяет минимизировать задержки и оперативно получать аналитические показатели. Каждое событие рассматривается как элемент бесконечной последовательности, а агрегаты обновляются инкрементально, что исключает необходимость ожидания полного набора данных. Этот подход позволяет обрабатывать миллионы событий в секунду и обеспечивает поддержку out-of-order событий, что критично для точности аналитики.

Предварительная агрегация заключается в раннем сворачивании данных на этапе их поступления. Суть метода состоит в том, чтобы на уровне edge-узлов или промежуточных сервисов формировать частичные агрегаты, которые затем передаются в центральное хранилище. Это позволяет значительно уменьшить объём передаваемых данных и снизить нагрузку на основную систему агрегации. Для повышения эффективности предварительной агрегации применяются иерархические структуры, в которых данные агрегируются сначала по секундам, затем по минутам, часам и дням. Такой подход обеспечивает возможность выполнения запросов различной временной глубины без необходимости пересчёта всех событий с нуля.

Использование оконных функций позволяет формировать агрегаты за фиксированные временные интервалы или по сессиям пользователей. В RTB-системах применяются непересекающиеся окна, скользящие окна и динамические сессионные окна, что обеспечивает гибкость при вычислении показателей в реальном времени. Особое внимание уделяется обработке опоздавших событий, когда события поступают не в хронологическом порядке. Для этого применяются механизмы отслеживания времени событий, допускаются периоды задержки обработки и реализуются стратегии ретроактивного пересчёта агрегатов, что позволяет поддерживать баланс между полнотой данных и оперативностью метрик.

Колоночные базы данных являются эффективным инструментом для аналитики больших потоков данных. В отличие от традиционных строковых СУБД, данные хранятся по столбцам, что улучшает компрессию, снижает объём считываемых данных и ускоряет выполнение запросов. Среди популярных решений для RTB-систем выделяются ClickHouse, Druid, BigQuery и Amazon Redshift. Они обеспечивают высокую пропускную способность вставки данных, поддержку распределённых вычислений и эффективную обработку аналитических запросов с минимальными задержками. Важным элементом является возможность использования материализованных представлений и агрегатных кубов, что позволяет заранее вычислять ключевые показатели и уменьшать время отклика на запросы аналитиков.

Для обеспечения масштабируемости и равномерного распределения нагрузки применяется партиционирование и шардирование данных. Временное партиционирование позволяет разделять данные по дням, неделям или месяцам, что упрощает архивирование и ускоряет выборку по времени. Хеш-шардирование распределяет данные между узлами на основе ключей, что позволяет минимизировать узкие места при высоких нагрузках. Гибридные стратегии, сочетающие временное и хеш-шардирование, позволяют обеспечить баланс между эффективностью хранения и быстродействием запросов. Кроме того, динамическое создание и поддержка партиций позволяет адаптировать систему к растущим объёмам данных без остановки работы платформы.

Применение описанных методов в комплексе позволяет RTB-системам достигать высокой пропускной способности, минимальных задержек при агрегации событий и точности аналитических показателей, что обеспечивает поддержку оперативного принятия решений в условиях высоких нагрузок.

В работе рассмотрены ключевые методы оптимизации агрегации событий в системах Real-Time Bidding при высоких нагрузках. Проведённый анализ показал, что традиционные подходы к обработке

данных оказываются недостаточно эффективными для обеспечения минимальных задержек и высокой пропускной способности в условиях миллионов событий в секунду.

Потоковая обработка данных позволяет обрабатывать события непрерывно, обеспечивая низкую латентность и поддержку событий вне хронологического порядка. Предварительная агрегация и использование материализованных представлений позволяют существенно снизить нагрузку на центральные хранилища и ускорить формирование аналитических метрик. Применение колоночных баз данных, а также партиционирование и шардирование данных обеспечивает масштабируемость системы и эффективность выполнения многомерных запросов.

Комплексное использование перечисленных подходов позволяет RTB-платформам достигать баланса между точностью вычислений, скоростью обработки и экономической эффективностью инфраструктуры. Полученные результаты могут быть использованы при проектировании и развитии высоконагруженных систем обработки данных в цифровой рекламе, а также служат основой для дальнейших исследований в области оптимизации агрегации потоковых событий.

Список использованных источников:

1. Zhu, F., & Iyer, B. Intermediation in Electronic Marketplaces: A Case of Real‑Time Bidding. Information Systems Research, 2018.

2. Hill, S., & Fulgoni, G. RTB Is Not Enough: Understanding the Role of Consumer Data in Digital Advertising. Journal of Advertising Research, 2015.

UDC 005.591.1

METHODS FOR OPTIMIZING EVENT AGGREGATION IN REAL-TIME BIDDING SYSTEMS UNDER HIGH LOADS

Antsiferova E.I.

Belarusian State University of Informatics and Radioelectronics
Minsk, Republic of Belarus

Davydova N.S. –PhD in Engineering, Associate Professor

Abstract. This paper presents an analytical review of methods for optimizing event aggregation in Real-Time Bidding systems under high loads. It examines the structural features of RTB logs and key processing challenges associated with high incoming data and significant volumes. It systematizes modern aggregation approaches, including stream processing, pre-aggregation methods, and the use of columnar data warehouses. A comparative analysis of the advantages and limitations of the solutions under consideration is provided. The results can be used in the design of scalable data processing systems for advertising platforms.

Keywords: Real-Time Bidding, RTB logs, event aggregation, streaming data processing, pre-aggregation, columnar databases, ClickHouse, Apache Kafka, high-load systems, scalability, data analytics