



ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ ПАТТЕРНА EXTERNAL PAYLOAD STORAGE В ИНТЕГРАЦИОННОМ КОНТУРЕ

Григорьев А. О. ORCID ID 0009-0008-7150-8676,
Фирсова С. А. ORCID ID 0000-0003-1831-684X,
Никишин М. Б. ORCID ID 0000-0002-1700-3676

*Федеральное государственное бюджетное образовательное учреждение высшего образования
«Национальный исследовательский Мордовский государственный университет
имени Н. П. Огарёва», Саранск, Российская Федерация, e-mail: agrigoryev13@yandex.ru*

Интеграционные контуры межведомственного обмена сообщениями при монолитном хранении тел документов в реляционной базе данных сталкиваются с ростом объема данных и снижением производительности журналирования и выборки метаданных. Разделение транзакционного реестра и крупных объектов полезной нагрузки во внешнем слое хранения рассматривается как перспективный архитектурный подход. Цель исследования – формализовать гибридную модель хранения полезной нагрузки сообщений по паттерну External Payload Storage, разработать методику количественной оценки объемов и времени операций записи и чтения (INSERT/SELECT) и подтвердить ее применимость экспериментально на промышленных инсталляциях. Объектом служила таблица учета сообщений интеграционного контура: до миграции тела документов размещались в полях реляционной таблицы, после – метаданные и ключи объектов в PostgreSQL, полезная нагрузка в объектном хранилище MinIO. На двух промышленных инсталляциях выполнен сравнительный эксперимент «до и после» с фиксацией объема строк и таблицы, времени записи и выборки метаданных, массовой журнальной выборки с фильтрацией по периоду. Установлено, что перенос полезной нагрузки во внешнее хранилище существенно уменьшает объем таблицы и среднюю длину строки, ускоряет операции записи и выборки метаданных; наибольший выигрыш достигается при журналировании без полного чтения документа. Сопоставление площадок разного масштаба подтвердило сохранение тенденций за счет снижения нагрузки на буферный кеш и перехода к индексному сканированию компактных строк; суммарное время полной записи и чтения сообщения остается сопоставимым с монолитной схемой. Предложенная модель и методика применимы при проектировании интеграционных контуров с преобладанием запросов к журналу и подтверждают целесообразность разделения транзакционной нагрузки между реляционной системой и объектным хранилищем.

Ключевые слова: интеграционный контур, полезная нагрузка, гибридное хранение, PostgreSQL, объектное хранилище, производительность

ASSESSMENT OF THE PERFORMANCE OF THE EXTERNAL PAYLOAD STORAGE PATTERN IN THE INTEGRATION CONTOUR

Grigorev A. O. ORCID ID 0009-0008-7150-8676,
Firsova S. A. ORCID ID 0000-0003-1831-684X,
Nikishin M. B. ORCID ID 0000-0002-1700-3676

*Federal State Budgetary Educational Institution of Higher Education
“National Research Mordovia State University named after N. P. Ogarev”,
Saransk, Russian Federation, e-mail: agrigoryev13@yandex.ru*

Integration environments for inter-agency message exchange that store document bodies monolithically in a relational database face growing data volumes and reduced performance of logging and metadata retrieval. Separating the transactional registry from large message payloads in an external storage layer is considered a promising architectural approach. The purpose of the study is to formalize a hybrid message payload storage model based on the External Payload Storage pattern, develop a methodology for quantitative assessment of storage volumes and read/write (INSERT/SELECT) operation times, and experimentally confirm its applicability on production installations. The object of study was the message accounting table of an integration environment: before migration, document bodies were stored in relational table fields; afterward, metadata and object keys remained in PostgreSQL and payloads in the MinIO object store. A before-and-after comparative experiment on two production installations recorded row and table volume, metadata write and read times, and bulk journal retrieval with period filtering. Moving payloads to external storage substantially reduces table volume and average row length and accelerates metadata write and read operations; the greatest benefit occurs in journal retrieval without full document reading. Comparison of installations at different scales confirmed consistent trends due to reduced buffer cache load and index scanning of compact rows; total time for full message write and read remains comparable to the monolithic scheme. The proposed model and methodology are applicable when designing integration environments dominated by message journal queries and confirm the feasibility of splitting transactional load between the relational system and object storage.

Keywords: integration environment, message payload, hybrid storage, PostgreSQL, object storage, performance

Введение

Интеграционные контуры обмена сообщениями нередко совмещают метаданные и тело XML в одной реляционной таблице. При росте документооборота увеличивается объем базы данных, замедляются INSERT и SELECT, особенно в сценариях журналирования и фильтрации без чтения полного XML. Цифровизация межведомственного взаимодействия предполагает устойчивое хранение истории обмена: оператору интеграционного контура необходимо быстро получать журнал сообщений, фильтровать записи по статусу и периоду, не загружая при этом полный XML каждого документа. Именно такие сценарии list/filter составляют основную долю обращений к таблице учета сообщений и определяют требования к производительности СУБД [1]. Ранее влияние накопления XML на объем реляционного контура систем межведомственного электронного взаимодействия (СМЭВ) рассматривалось авторами в задаче резервного копирования [2]. Архитектурной альтернативой монолитному хранению является гибридная архитектура, в которой можно использовать композицию паттернов Claim Check и External Payload Storage (EPS), где в транзакционном хранилище остается «квитанция» (ключ), а полезная нагрузка выносится во внешнее хранилище соответственно. Паттерн Claim Check [3, 4] отвечает за логику «выдал квитанцию – забрал по квитанции»: в БД хранится только ключ, а данные временно вынесены. Паттерн EPS [5, 6] задает конкретное решение – в какое внешнее хранилище выносить данные и как ими управлять. Близкий прием – отделение реестра от тел объектов – рассматривается в работах по организации хранения данных [7, 8]. В отличие от монолитного хранения и частичного выноса через TOAST PostgreSQL [9], EPS явно разграничивает ответственность контуров и позволяет независимо масштабировать объектное хранилище [10]. В связи с этим исследование возможности применения гибридной модели с паттерном EPS в высоконагруженных системах с целью ускорения операций чтения/записи представляется актуальным.

Цель исследования – формализовать гибридную модель хранения полезной нагрузки сообщений по паттерну External Payload Storage, разработать методику количественной оценки объемов и времени операций записи и чтения (INSERT/SELECT) и подтвердить ее применимость экспериментально на промышленных инсталляциях.

Материалы и методы исследования

Объект – таблица `smev_messages` интеграционного контура. До EPS XML размещались в полях `a_xml`, `a_service_message`, `a_generated_message`; после миграции в PostgreSQL остаются идентификаторы, статусы, даты и ключи объектов, а `payload` хранится в MinIO [11]. Эксперимент проводился на БД № 1 (~89 ГБ) и БД № 2 (~1768 ГБ).

Для количественной оценки скорости операций чтения/записи применялись следующие методы сбора данных. Объемные характеристики определялись средствами PostgreSQL (параметры `pg_database_size`, `pg_total_relation_size`, `pg_column_size`) [12]. Планы выполнения и нагрузка на буферный кеш анализировались с помощью EXPLAIN (ANALYZE, BUFFERS). Временные характеристики операций записи и чтения измерялись с помощью Apache JMeter [13] на выборке $N = 1000$ операций; для каждой серии фиксировались минимальное, среднее и максимальное время. Массовая выборка моделировала запрос оператора с фильтром по диапазону дат и ограничением $M = 1000$ строк:

$$L = L_m + L_p, \quad (1)$$

где L_m – служебные поля; L_p – XML-поля (`pg_column_size`).

Доля `payload` в объеме БД:

$$d_p = \left(\frac{V_p}{V_{db}} \right) \times 100\%, \quad (2)$$

где V_p – объем XML-полей; V_{db} – объем базы данных.

Сокращение таблицы после EPS:

$$\delta_V = \frac{V_{tbl}^0 - V_{tbl}^1}{V_{tbl}^0} \times 100\%, \quad (3)$$

где V_{tbl}^0 , V_{tbl}^1 – `pg_total_relation_size` до и после миграции.

Среднее время INSERT ($N = 1000$):

$$T_{INS}^0 = \frac{1}{N} \sum_{i=1}^N t_i, \quad (4)$$

где t_i – время i -й записи в монолитной схеме; после EPS:

$$T_{INS}^1 = T_{INS}^m + T_{PUT}, \quad (5)$$

где T_{INS}^m – INSERT метаданных; T_{PUT} – PUT в S3.

Коэффициент ускорения записи в СУБД:

$$K_{INS} = \frac{T_{INS}^0}{T_{INS}^m}. \quad (6)$$

Также для выборки метаданных вводился коэффициент K_{SEL} как отношение среднего времени SELECT до и после внедрения EPS:

$$K_{SEL} = \frac{T_{SEL}^0}{T_{SEL}^1}. \quad (7)$$

В монолитной схеме по формуле (4) измеряется одно время записи T_{INS}^0 : INSERT строки вместе с XML. После EPS это время разделяется: T_{INS}^m – только INSERT метаданных в PostgreSQL, без обращения к объектному хранилищу, T_{PUT} – только PUT XML в MinIO. Полный цикл записи сообщения является суммой этих величин T_{INS}^1 (формула (5)). В формуле (6) коэффициент K_{INS} относится исключительно к T_{INS}^m и не характеризует полный цикл T_{INS}^1 . Сопоставление полного цикла с монолитной записью выполняется по T_{INS}^1 и T_{INS}^0 , а не по K_{INS} .

Запись в случае использования EPS включает три последовательных шага: прием сообщения, загрузка XML во внешнее хранилище и INSERT компактной строки метаданных в PostgreSQL. Согласованность двух шагов обеспечивает Storage Adapter: при сбое записи в объектное хранилище транзакция в СУБД не фиксируется. Такой прикладной порядок согласуется с паттерном Claim Check и опирается на модели транзакционной фиксации [14] и восстановления [15]. Чтение реализовано в двух сценариях: сценарий А – SELECT только метаданных из PostgreSQL (журналирование / list-filter); сценарий В – SELECT метаданных с последующим GET объекта из S3 для формирования полного сообщения. Архитектура EPS и соответствующий процесс выполнения операций чтения/записи наглядно представлены на рис. 1 и 2.

Таким образом, методика сочетает объемный анализ реляционного слоя, инструментальный замер латентности операций и сравнительную оценку на двух площадках, что позволяет отделить эффект сокращения длины строки от влияния масштаба данных.

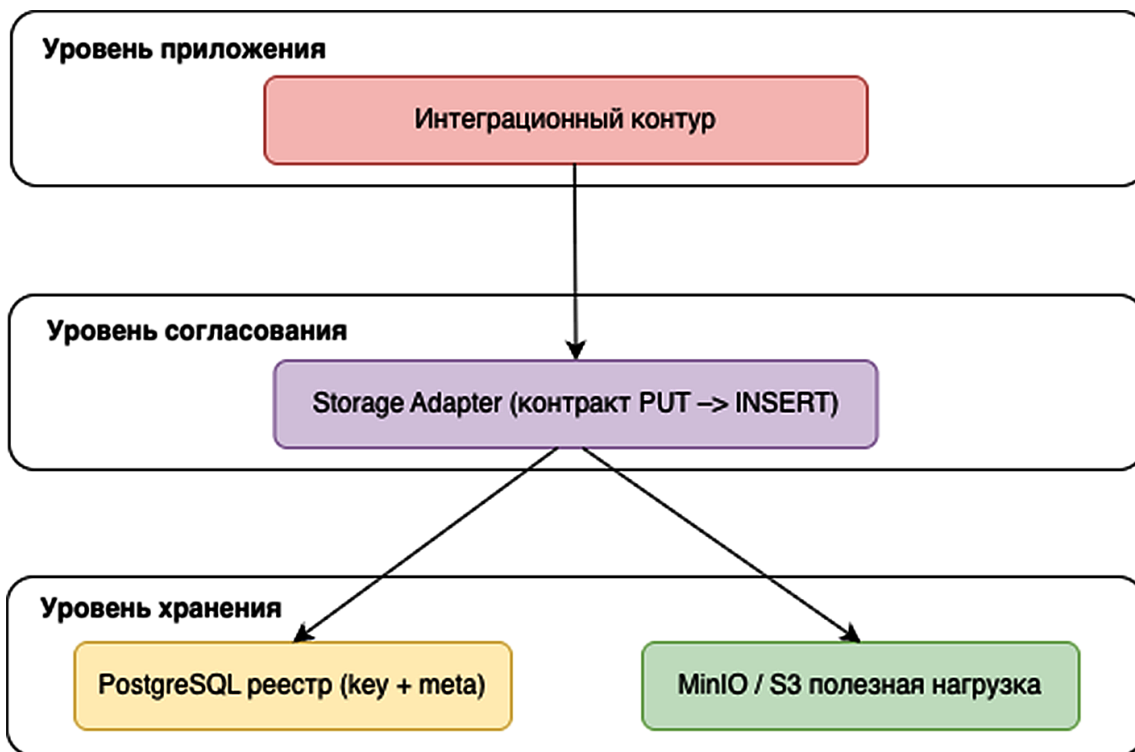


Рис. 1. Архитектура паттерна External Payload Storage
Примечание: составлен авторами по результатам данного исследования

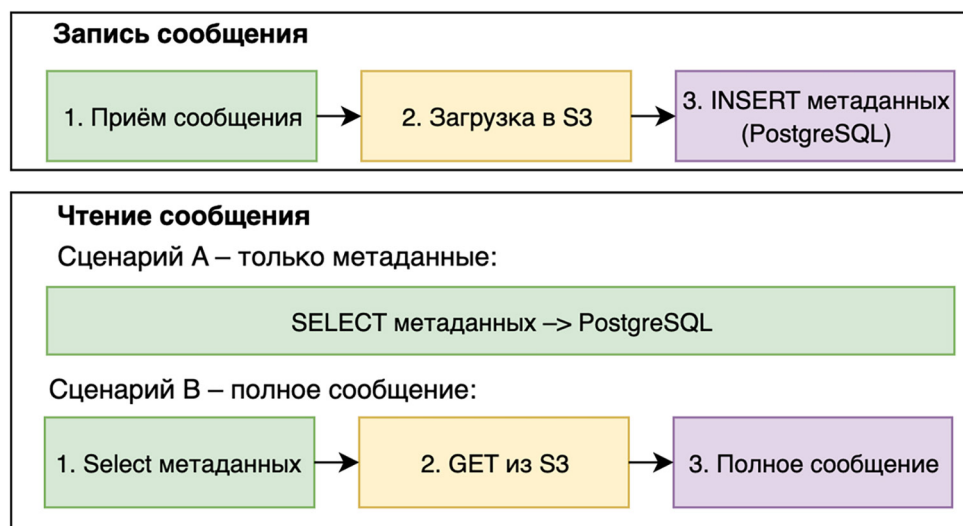


Рис. 2. Контракты чтения/записи при использовании EPS
Примечание: составлен авторами по результатам данного исследования

Результаты исследования и их обсуждение

В рамках данного исследования под гибридной моделью хранения полезной нагрузки сообщений в работе понимается разделение данных сообщения на два связанных контура:

1) транзакционный реестр в PostgreSQL – компактная строка метаданных сообщения (идентификаторы, статусы, временные метки, атрибуты маршрутизации) и ключ объекта полезной нагрузки;

2) внешнее объектное хранилище (S3-совместимое [16], MinIO) – тело сообщения (XML/payload) как независимый объект по этому ключу.

Связь между контурами однозначная: каждой строке реестра соответствует ровно один объект полезной нагрузки. Согласованной запись считается лишь тогда, когда успешно завершены оба шага контракта – размещение объекта во внешнем хранилище (PUT) и фиксация строки реестра (INSERT/UPDATE). Если сбой происходит на стороне объектного слоя, транзакция в СУБД не подтверждается. Для чтения предусмотрены два сценария. В первом из СУБД выбираются только метаданные (журнал, list/filter). Во втором к метаданным добавляется извлечение payload через GET, и сообщение собирается целиком.

От монолитного хранения XML в строке таблицы и от TOAST PostgreSQL предложенная модель External Payload Storage отличается тем, что крупные объекты оказываются вне СУБД. Реляционный контур в этом случае ведет учет транзакций и хранит ключ, обеспечивающий ссылку на внешний объект.

Алгоритм, представленный на рис. 3, задает методику количественной оценки объемов данных и времени выполнения операций чтения/записи и включает следующие этапы.

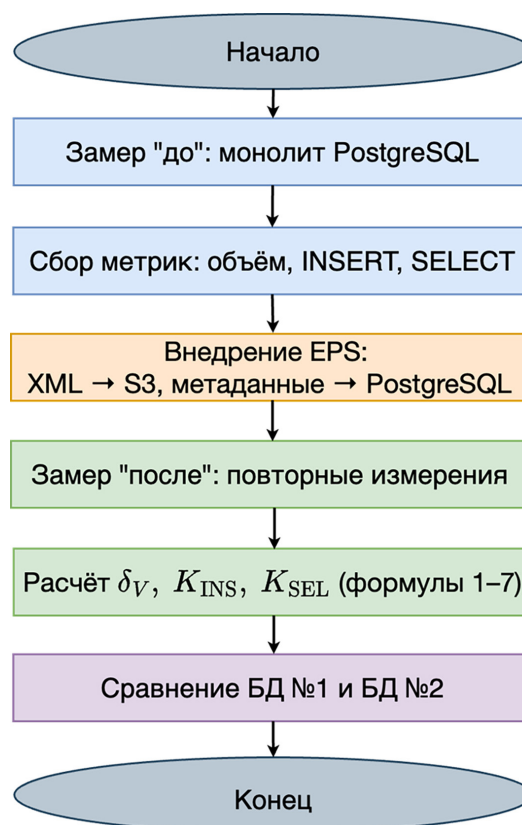


Рис. 3. Алгоритм экспериментальной оценки паттерна EPS
Примечание: составлен авторами по результатам данного исследования

Опишем каждый из этапов предложенного алгоритма.

Этап 1. Фиксация исходного состояния монолитной схемы: объем базы V_{db} , объем таблицы сообщений V_{tbl}^0 , средняя длина строки L , доля полезной нагрузки d_p .

Этап 2. Замер временных характеристик «до EPS»: среднее время INSERT (T_{INS}^0), SELECT метаданных (T_{SEL}^0) и массовой журнальной выборки (T_{list}^0) на выборке $N=1000$ операций и $M=1000$ строк.

Этап 3. Миграция полезной нагрузки во внешнее хранилище и переход к гибридной схеме (метаданные и ключ объекта – в PostgreSQL, payload – в MinIO).

Этап 4. Повторный замер тех же показателей «после EPS»: объем таблицы V_1 , время INSERT метаданных T_{INS}^m и полного цикла записи $T_{INS}^1 = T_{INS}^m + T_{PUT}$, время SELECT метаданных T_{SEL}^1 и журнальной выборки T_{list}^1 .

Этап 5. Расчет относительных эффектов по формулам (1)–(7): сокращение объема δ_v , коэффициенты ускорения K_{INS} и K_{SEL} .

Этап 6. Сопоставление результатов на БД № 1 и БД № 2.

Таким образом, рис. 3 отражает не архитектуру хранения, а порядок измерений и расчетов, обеспечивающий воспроизводимое сравнение монолитной и гибридной схем.

В результате проведенных вычислений были сделаны следующие выводы.

На БД № 1 $d_p = (28/89) \cdot 100\% \approx 31,5\%$; на БД № 2 – $35,6\%$. После EPS $\delta_v \approx 90,3\%$

и $90,5\%$; объем таблицы снизился с 31 до 3 ГБ и с 696 до 66 ГБ (рис. 4, таблица). L уменьшился с 18–42 до 0,3–0,8 КБ; общий V_{db} – с 89 до 61 ГБ и с 1768 до 1071 ГБ.

На БД № 2 сокращение таблицы сообщений составило 630 ГБ ($696 \rightarrow 66$), а сокращение всей базы данных – 697 ГБ ($1768 \rightarrow 1071$). Дополнительные ≈ 67 ГБ объясняются уменьшением структур, связанных с XML-полями: внутренней TOAST-таблицы и индексов / служебных объектов отношения после удаления крупных атрибутов, а также отражением освобожденного места в показателе `pg_database_size` после миграции.

Результаты операции INSERT: $T_{INS}^0 = 300$ мс $\rightarrow T_{INS}^m = 25$ мс, $K_{INS} = 12$ (рис. 5); полное $T_{INS}^1 \approx 105$ мс ($T_{PUT} \approx 80$ мс). Результаты операции SELECT метаданных: $T_{SEL}^0 = 80$ мс $\rightarrow T_{SEL}^1 = 7,5$ мс, $K_{SEL} \approx 10,7$ (рис. 6). Журнальная выборка ($M=1000$): $7,8$ с $\rightarrow 0,72$ с; shared buffers: $8420 \rightarrow 1180$ ($K_{buf} \approx 7,1$). Полное чтение сообщения (SELECT + GET) осталось сопоставимым: $45\text{--}130$ мс.

Сопоставление двух площадок показывает сохранение тенденций при разном масштабе данных. На БД № 2 абсолютные значения T_{INS}^0 и T_{SEL}^0 выше, однако относительный выигрыш после внедрения EPS не снижается: $K_{INS} \approx 13,1$, $K_{SEL} \approx 15$. Это подтверждает, что эффект обусловлен не только уменьшением L , но и снижением объема данных, считываемых из буферного кеша PostgreSQL при каждой транзакции.

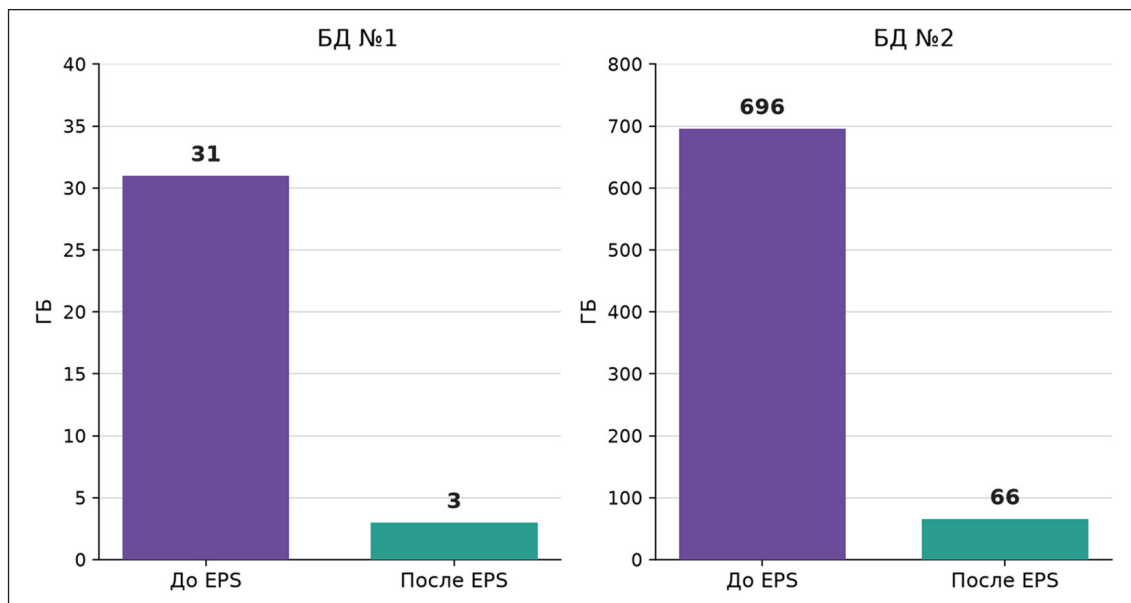


Рис. 4. Объем таблицы сообщений до и после внедрения EPS

Примечание: составлен авторами по результатам данного исследования

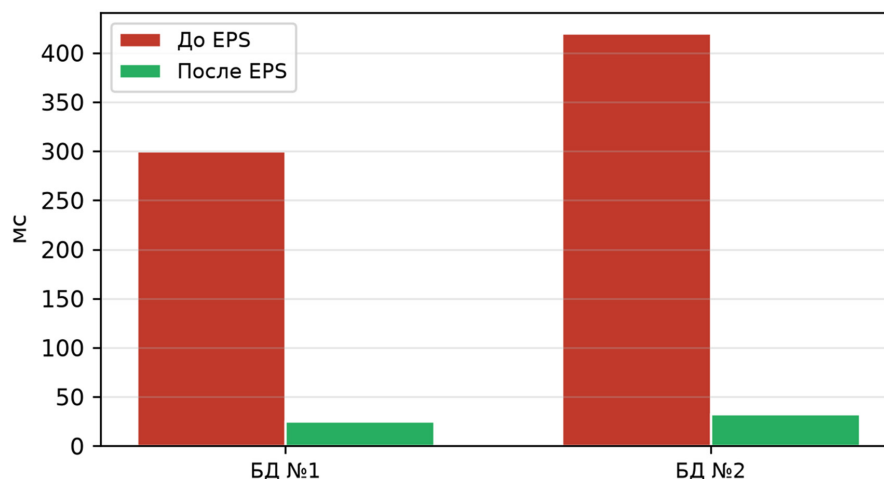


Рис. 5. Среднее время операции вставки (INSERT) в PostgreSQL до и после EPS
Примечание: составлен авторами по результатам данного исследования

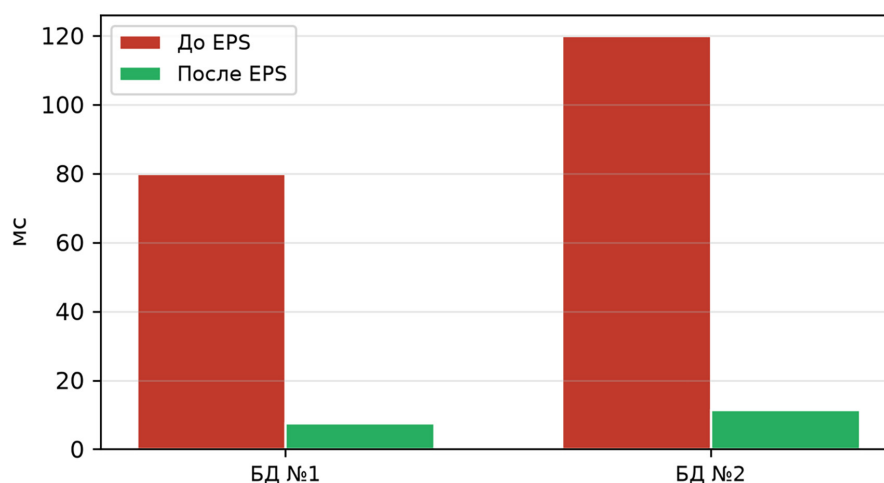


Рис. 6. Среднее время операции чтения (SELECT) метаданных до и после EPS
примечание: составлен авторами по результатам данного исследования

Полученные результаты эксперимента

Показатель	БД № 1	БД № 2
V_{db} , ГБ	89 → 61	1768 → 1071
Таблица, ГБ	31 → 3	696 → 66
δ_p , %	90,3	90,5
K_{INS} / K_{SEL}	12 / 10,7	13,1 / 15
T_{list} , с	7,8 → 0,72	12,5 → 0,95

Примечание: составлена авторами на основе полученных данных в ходе исследования.

При переходе на EPS основная транзакционная нагрузка остается в PostgreSQL, тогда как тела сообщений выносятся в S3-совместимое хранилище. Расчетные и экспериментальные значения по формулам (1)–(7) совпадают по порядку величин.

Для исследованного профиля нагрузки – журнальные list/filter-запросы к метаданным без полного чтения XML – гибридная схема на двух площадках предпочтительнее монолитного хранения. Обобщение на контуры с иной долей полного чтения Payload требует отдельной оценки трафика.

Время записи T_{INS}^1 складывается не только из INSERT в СУБД, но и из сетевого PUT; при интенсивном потоке входящих сообщений узким местом становятся пропускная способность объектного слоя и степень параллелизма Storage Adapter. Для полного просмотра сообщения задержка в основном определяется GET-запросом и для работы оператора остается в допустимых пределах. Так, EPS наиболее оправдан, когда система чаще ищет и фильтрует метаданные, чем выгружает полные XML-тела.

Заключение

На двух исследованных промышленных инсталляциях и для принятого профиля нагрузки (преобладание журнальных list/filter-запросов к метаданным без полного чтения XML) паттерн EPS показал устойчивый эффект: объем табличной части сообщений сократился примерно на 90%, операции INSERT и выборка метаданных ускорились в 8–15 раз, при этом полное чтение сообщения по времени осталось сопоставимым с монолитной схемой. Полученные результаты подтверждены для указанных площадок и профиля запросов; перенос выводов на контуры с иной долей полного чтения payload требует отдельной экспериментальной оценки.

Для промышленного внедрения необходимо зафиксировать правила формирования ключей объектов, контролировать связь «метаданные ↔ объект в S3» и отслеживать длительность PUT/GET. Предложенный алгоритм оценки удобен на этапе модернизации: достаточно снять показатели «до / после» на репрезентативной выборке и вычислить δ_V , K_{INS} и K_{SEL} по приведенным формулам. В дальнейшем планируется проверить EPS при гео-распределенном объектном хранилище и асинхронной репликации Payload.

Список литературы

1. Yu M., Lin Z., Sun J., Zhou R., Jiang G., Huang H., Zhang S. TencentCLS: The Cloud Log Service with High Query Performances // Proceedings of the VLDB Endowment. 2022. Vol. 15. Is. 12. P. 3472–3482. URL: <https://www.vldb.org/pvldb/vol15/p3472-yu.pdf> (дата обращения: 20.08.2026). DOI: 10.14778/3554821.3554837.
2. Григорьев А. О., Фирсова С. А., Жарков Р. А. Архитектура резервного копирования данных СМЭВ с выносом XML-сообщений в S3-совместимое объектное хранилище: постановка задачи и экспериментальная оценка // Инженерный вестник Дона. 2026. № 7. URL: <https://www.ivdon.ru/rumagazine/archive/n7y2026/11475> (дата обращения: 01.08.2026).
3. Claim-Check pattern // Azure Architecture Center [Электронный ресурс]. URL: <https://learn.microsoft.com/en-us/azure/architecture/patterns/claim-check> (дата обращения: 03.08.2026).
4. Cloud design patterns, architectures, and implementations // Amazon Web Services URL: <https://docs.aws.amazon.com/prescriptive-guidance/latest/cloud-design-patterns/claim-check.html> (дата обращения: 03.08.2026).
5. External Storage // Temporal Docs [Электронный ресурс]. URL: <https://docs.temporal.io/external-storage> (дата обращения: 03.08.2026).
6. External Payload Storage // Conductor Docs [Электронный ресурс]. URL: <https://docs.conductor-oss.org/documentation/advanced/externalpayloadstorage.html> (дата обращения: 08.08.2026).
7. Лотов А. А., Савицкий Е. Н., Перевалова М. Н., Павлова Е. А. Системы организованного хранения и обработки файлов // Математическое и информационное моделирование: материалы Всероссийской конференции молодых ученых (г. Тюмень, 22–30 апреля 2025 г.). Вып. 23. Тюмень: ТюмГУ-Press, 2025. С. 230–235. URL: <https://elibrary.ru/item.asp?id=82747631> (дата обращения: 01.06.2026).
8. Подвесовский А. Г., Коростелев Д. А., Захарова А. А. Организация хранения данных функционирования объектов киберфизических систем // Программные продукты и системы. 2024. № 3. С. 334–343. URL: <https://elibrary.ru/item.asp?id=71311331> (дата обращения: 01.06.2026). DOI: 10.15827/0236-235X.147.334-343.
9. PostgreSQL Documentation: TOAST // PostgreSQL Global Development Group [Электронный ресурс]. URL: <https://www.postgresql.org/docs/current/storage-toast.html> (дата обращения: 03.08.2026).
10. Полянцева К. А., Комлев А. В., Городничев М. Г. Архитектура распределенной системы хранения и обработки больших данных на основе Apache Ozone и Argo Workflows // Известия Кабардино-Балкарского научного центра РАН. 2026. Т. 28. № 2. С. 34–50. URL: <https://www.elibrary.ru/item.asp?id=89328448> (дата обращения: 03.06.2026). DOI: 10.35330/1991-6639-2026-28-2-34-50.
11. MinIO AIStor Documentation // MinIO [Электронный ресурс]. URL: <https://min.io/docs/minio/linux/index.html> (дата обращения: 03.08.2026).
12. PostgreSQL Documentation // PostgreSQL Global Development Group [Электронный ресурс]. URL: <https://www.postgresql.org/docs/> (дата обращения: 03.08.2026).
13. Apache JMeter User's Manual // Apache Software Foundation. [Электронный ресурс]. URL: <https://jmeter.apache.org/usermanual/index.html> (дата обращения: 03.08.2026).
14. Демиш В. О., Пищик Б. Н. Моделирование обмена транзакционными данными в корпоративных системах // Вестник НГУ. Серия: Информационные технологии. 2020. Т. 18. № 1. С. 5–15. URL: <https://elibrary.ru/item.asp?id=42562120> (дата обращения: 05.08.2026). DOI: 10.25205/1818-7900-2020-18-1-5-15.
15. Нуркаев Р. Р., Пивоваров В. В., Хабибуллин Р. М., Хасанов А. Р. SAGA: управление долговременными транзакциями в распределенных системах // Инновации и инвестиции. 2023. № 6. С. 175–178. URL: <https://elibrary.ru/item.asp?id=54159794> (дата обращения: 06.08.2026).
16. Amazon Simple Storage Service (S3) User Guide // Amazon Web Services. [Электронный ресурс]. URL: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html> (дата обращения: 01.06.2026).

Конфликт интересов: Авторы заявляют об отсутствии конфликта интересов.

Conflict of interest: The authors declare that there is no conflict of interest.

Финансирование: Авторы заявляют об отсутствии внешнего финансирования.

Financing: The research was performed without external funding.