

ELASTICSEARCH

второй шанс на метриках

01.tech

Maybe this time
it will be fast...

- too many shards
- expensive queries
- unbounded cardinality
- just one more field...



SMALL
BUCKETS
BIG
ANSWERS

index
bloat

high
cardinality

too many
fields

same
mistakes?

- ☒ Ingest
- ☒ Store
- ☒ Downsample
- ☐ Profit?

Немного о себе

Денис Габайдулин

- Platform developer в компании 01.tech
- Ранее работал:
 - Ozon (поисковый движок)
 - Yandex (YQL)
 - Odnoklassniki (bigdata)



О компании

- Делаем решения для Entertainment Industry с **2016 года**
- Растем на **5%** каждый месяц
- MAU **2M+**
- DAU **200K+**
- Клиенты со всего мира

О масштабе метрик

- До 2 млн документов в секунду на трех кластерах
- 100+ датастримов

О масштабе метрик

Один из кластеров

5

триллион

документов

140

ТВ

диск

25k

шардов

primary & replicas

О чем доклад

- Насколько **Elasticsearch** годится для метрик?
- Устройство time-series data streams (TSDS)
- Практические советы по конфигурации
- Истории из прода

Требования

- Эффективно хранить метрики за 2 года (columnar store)
- Масштабирование
- Отказоустойчивость
- Доступность

Почему Elasticsearch

- Уже хранили логи и трейсы
- Поддержка columnar store
- Эластичность + HA (репликация и шардирование)
- Opensource
- Написан на Java

Критерии

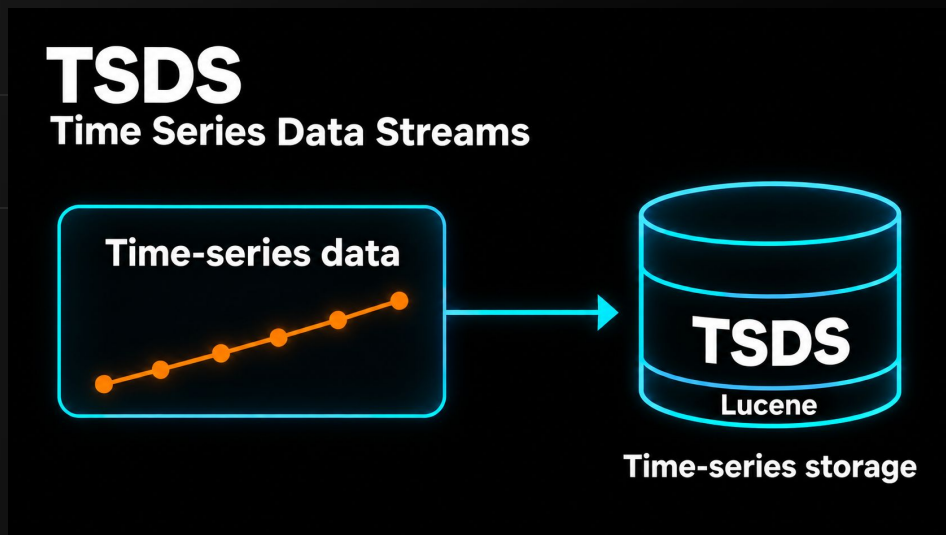
- Эффективность хранения (cost)
- Шардирование (scaling out)
- Репликация (HA)
- Downsampling (cost)
- Надежность/эксплуатация

Прогресс 0%

Стек

Что предлагает Elasticsearch

Фундамент

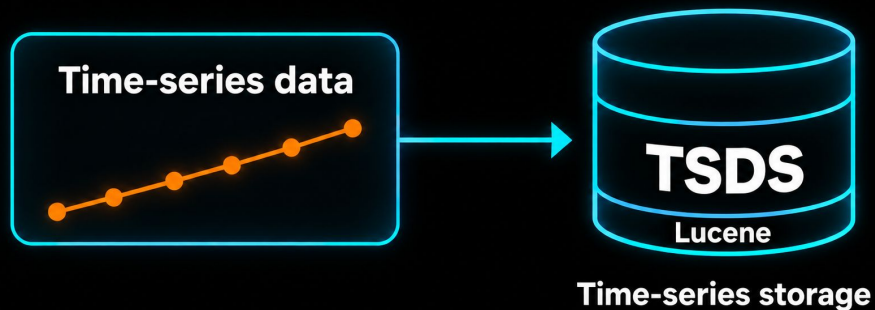


Что предлагает Elasticsearch

Фундамент

TSDS

Time Series Data Streams



Фичи

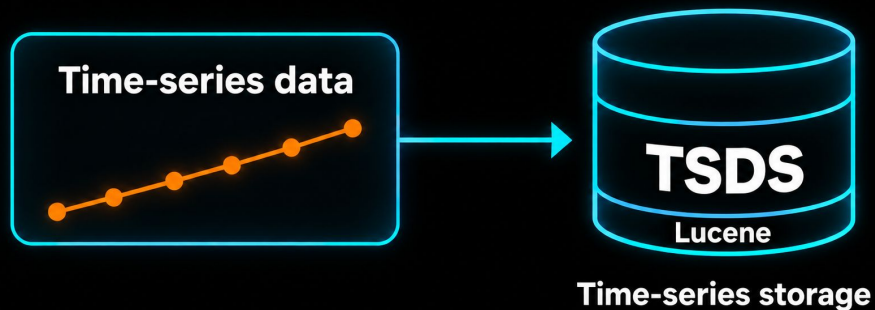
01 Компрессия данных

Что предлагает Elasticsearch

Фундамент

TSDS

Time Series Data Streams



Фичи

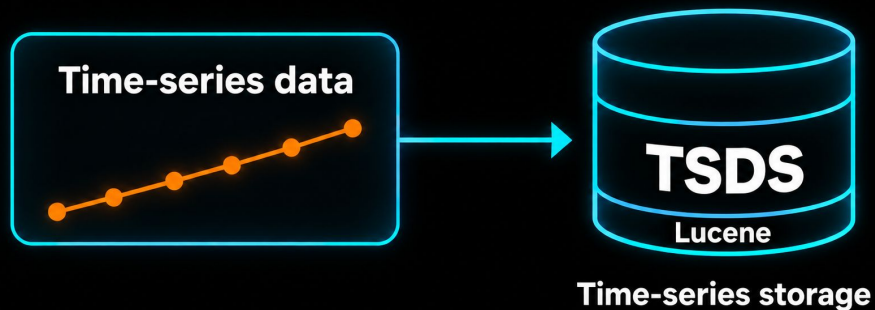
- 01 Компрессия данных
- 02 Языки запросов ESIQL, PromQL

Что предлагает Elasticsearch

Фундамент

TSDS

Time Series Data Streams



Фичи

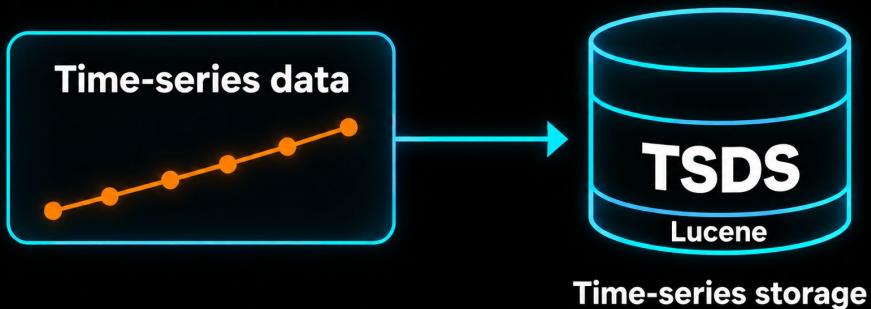
- 01 Компрессия данных
- 02 Языки запросов ESIQL, PromQL
- 03 Downsampling (30 секунд → 5 минут)

Что предлагает Elasticsearch

Фундамент

TSDS

Time Series Data Streams



Фичи

- 01 Компрессия данных
- 02 Языки запросов ESIQL, PromQL
- 03 Downsampling (30 секунд → 5 минут)
- 04 Динамическая схема

Устройство TSDS

Что такое TSDS

- Удобная модель: **control plane** + **data plane** для time-series данных

TSDS insertion

1. Send to the data stream

Document

@timestamp	10:42
host	pay-01
latency_ms	12

TSDS

metrics-payments

TSDS insertion

2. Select index by timestamp

Document

@timestamp	10:42
host	pay-01
latency_ms	12

TSDS

metrics-payments

Backing indices

Index 1

[09:00, 10:00)

Index 2

[10:00, 11:00)

Index 3

[11:00, 12:00)

@timestamp

TSDS insertion

3. Write to a shard by dimensions

Document

@timestamp	10:42
host	pay-01
latency_ms	12

TSDS

metrics-payments

@timestamp

Backing indices

Index 1

[09:00, 10:00)

Shard 0

Shard 1

Index 2

[10:00, 11:00)

host: pay-01

Shard 0

Shard 1

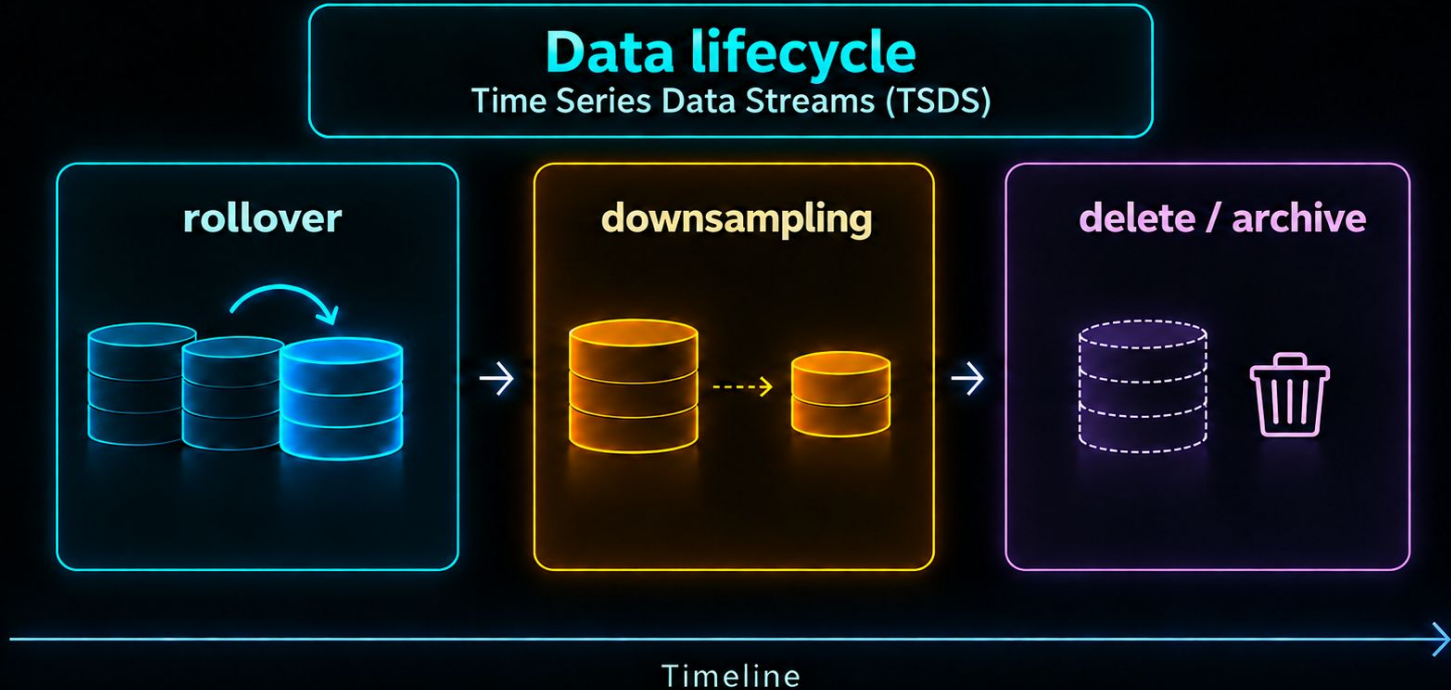
Index 3

[11:00, 12:00)

Shard 0

Shard 1

Control plane TSDS



Data plane TSDS

- Data plane: Lucene + специализированные TSDB encodings/codecs
- Данные хранятся в columnar store

Проверим как хранятся данные

► Persisted value bytes counted once per field

▼ Physical (compressed) bytes by Lucene type

CATEGORY	PHYSICAL BYTES	FIELD-ATTRIBUTED	SHARED / RESIDUAL	FILE PARTS	COMMIT SHARE
Inverted index	54.84 MiB (57,507,085 B)	54.84 MiB (57,506,608 B)	477 B	4	2.42%
Stored fields	15.18 MiB (15,912,781 B)	0 B	15.18 MiB (15,912,781 B)	3	0.67%
Doc values	2.09 GiB (2,247,423,147 B)	1.9 GiB (2,044,758,221 B)	193.28 MiB (202,664,926 B)	4	94.62%
Bloom filter	4 MiB (4,194,518 B)	4 MiB (4,194,304 B)	214 B	2	0.18%
Field infos	504.8 KiB (516,920 B)	0 B	504.8 KiB (516,920 B)	1	0.02%
Segment metadata	837 B	0 B	837 B	1	0.00%
Commit metadata	435 B	0 B	435 B	1	0.00%
Other/shared	47.35 MiB (49,645,422 B)	0 B	47.35 MiB (49,645,422 B)	2	2.09%
Latest commit total	2.21 GiB (2,375,201,145 B)	1.96 GiB (2,106,459,133 B)	256.29 MiB (268,742,012 B)		100.00%

Эффективность хранения

Сжатие данных

- Насколько хорошо работает компрессия данных?

Сжатие **данных**

Какие данные у нас

- Текстовые атрибуты: строки
- Метрики: long, double
- В документе обычно может быть 20-40 полей (атрибуты и метрики)

```
{  
  "@timestamp": "2026-10-07T18:30:00.000+03:00",  
  "_metric_names_hash": "51fe2ad455579324",  
  "attributes.area": "nonheap",  
  "attributes.id": "CodeHeap 'non-nmethods'",  
  "attributes.instrumentation_source": "jvm",  
  "data_stream.namespace": "platform",  
  "data_stream.type": "metrics",  
  "labels.host_group": "balancer_sa_stage",  
  "labels.service_namespace": "platform",  
  "metrics.jvm.memory.committed": 2555904,  
  ...  
  "service.environment": "staging",  
  "service.language.name": "java",  
  "service.name": "one-lb-agent"  
}
```

Сжатие данных

Версия 9.3.0

10-13

bytes/document

6-7

bytes/metric point

Сжатие данных

Версия 9.5.0

2-4

bytes/document

1.8

bytes/metric point

Сжатие **данных**

Версия 9.5.0

- **Убрали** служебные поля `_id`, `_seq_no`



Улучшили компрессию

Delta encoding

Values (on disk)

Series A

1000	1001	1002	1003
------	------	------	------

Delta

-	+1	+1	+1
---	----	----	----

Fixed-size block

1000	1001	1002	1003
-	+1	+1	+1

1 bit/delta

Delta encoding

Values (on disk)



Series boundary

Delta



Fixed-size block
(spans boundary)



12 bits/delta

Large jump



Poor compression

Delta encoding

Values (on disk)



Delta



Fixed-size block
(spans boundary)



12 bits/delta

Large jump



With SplitDelta

Values (on disk)



Delta



SplitDelta blocks
(respect boundaries)



1 bit/delta

1 bit/delta

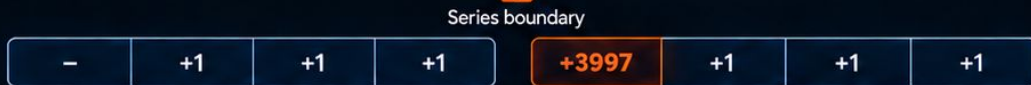


Delta encoding

Values (on disk)



Delta



Fixed-size block
(spans boundary)



12 bits/delta

Large jump

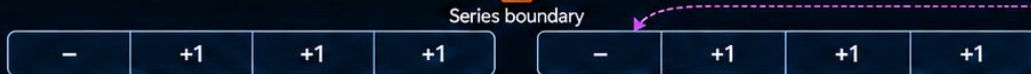


With SplitDelta

Values (on disk)



Delta



SplitDelta blocks
(respect boundaries)



1 bit/delta

1 bit/delta



Timestamp reduction
16 bits per value → 4 bits per value

Сжатие данных

Что сжимается не так хорошо?

- Числа с плавающей точкой (ALP не всегда спасает)

FIELD	TYPE	CODEC / MODEL	ENCODING	VALUES	BITS/VALUE
metrics.in_app_performance_sum	sorted numeric	ES95 TSDB	alpDouble > delta > offset > gcd > bitPack	4,158,045	50.6

ALP Adaptive Lossless Floating-Point Compression

Сжатие **данных**

Что сжимается **не так хорошо?**

Floating-point с неоднородной точностью

0.008

3 знака



0.103

3 знака



0.00000064

8 знаков



1.363

3 знака



32.011

3 знака

Сжатие **данных**

Что сжимается **не так хорошо?**



Случайные целые числа

Из широкого диапазонов значений

`random: 0 ...  $2^{64} - 1$`



Непредсказуемость

Значения последовательности практически невозможно предсказать или аппроксимировать



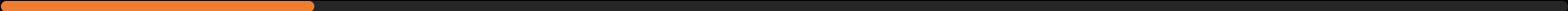
Высокий разброс

Большие и хаотичные дельты между соседними значениями исключают эффективный delta-encoding

Критерии

- Эффективность хранения (cost) ✓
- Шардирование (scaling out)
- Репликация (HA)
- Downsampling (cost)
- Надежность/эксплуатация

Прогресс 20%

A horizontal progress bar is located at the bottom of the slide. It consists of a dark grey line with an orange segment on the left, representing the 20% progress.

Организация данных

Организация данных

- Эффективное чтение данных
- Масштабирование

Data stream

metrics-otelcol.v1-devops.prod

Index 1

Shard 1

Index 2

Shard 1

Shard 2

Index 3

Shards

1

...

N

Each index: 1-N shards

Организация **данных**

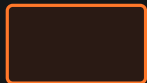
Две независимые задачи

01 · ЛОГИЧЕСКАЯ ОРГАНИЗАЦИЯ

Группировка метрик по
датастримам



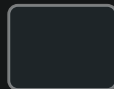
департамент



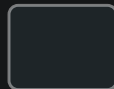
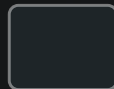
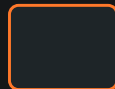
scope.name

02 · ФИЗИЧЕСКАЯ ОРГАНИЗАЦИЯ

Конфигурация шардов



1-5 primary shards

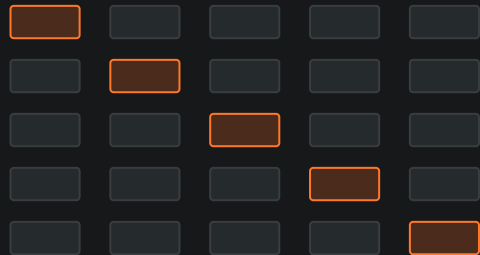


Сначала режем данные логически, потом подбираем физическую раскладку.

Зачем группировать метрики

Эффективное чтение данных

БЕЗ ГРУППИРОВКИ



FROM metrics-otelcol*

С ГРУППИРОВКОЙ

devops.prod

payments.prod

platform.prod

FROM metrics-otelcol*devops.prod*



Группировка метрик

- Сужаем пространство поиска
 - FROM metrics-otelcol*devops.prod*
- Ограничиваем число полей в схеме одного датастрима

Две стратегии группировки

01

По департаментам

ownership / ACL / quotas

DevOps → data stream

Payments → data stream

Platform → data stream

Метрики сервисов одного владельца лежат компактно и управляются вместе.

02

По scope.name

platform metrics

GRPC

JVM

HTTP

Одинаковые платформенные метрики собираем вместе независимо от владельца.

department - на «кто владеет?»

scope.name - на «что это за класс метрик?»

Группировка по департаментам

Бизнесовые метрики локализуем в отдельном датастриме

CDP

service-a

service-b

service-c

→ metrics-otelcol.v1-cdp.prod

Payments

service-d

service-e

→ metrics-otelcol.v1-payments.prod

Platform

service-f

service-g

service-h

→ metrics-otelcol.v1-platform.prod

ЧТО ПОЛУЧАЕМ

01 Компактная локальность

метрики сервисов одного владельца лежат вместе

02 Права доступа

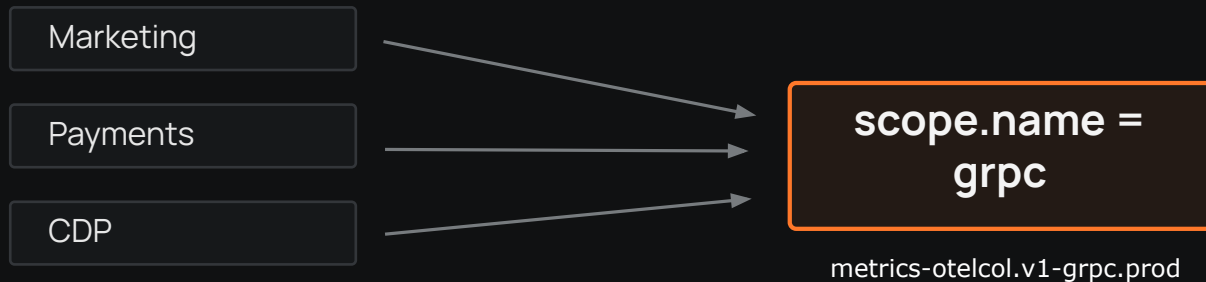
ACL можно выдавать на data stream / pattern

03 Ограничения

квоты и lifecycle проще задавать на группу

Группировка по `scope.name`

Платформенные метрики режим поперек оргструктуры



ЗАЧЕМ

Поиск

запрос идет сразу в датастрим с нужным классом метрик

Дашборды

одни и те же platform metrics видны целиком по всем сервисам

Пример: (G)RPC-метрики всех сервисов - в одном датастриме

Конфигурация шардов

Кластера

Наша конфигурация

5-7 нод

Кластера

Наша конфигурация

5-7 нод

2 x 32 cores

Кластера

Наша конфигурация

5-7 нод

2 x 32 cores

1 TB RAM

Шарды

Наша конфигурация

≤ 50 GB

размер шарда

Шарды

Наша конфигурация

≤ 50 GB

размер шарда

≤ 1 B docs

документов в шарде

(нужен патч)

Шарды

Наша конфигурация

≤ 50 GB

размер шарда

≤ 1 B docs

документов в шарде

(нужен патч)

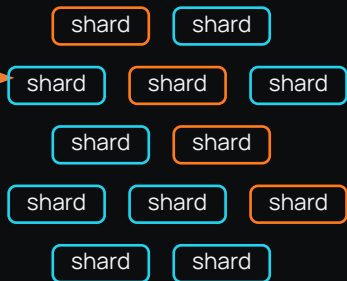
Цель Уменьшить общее количество шардов

Шарды

Мелкие шарды - но большие проблемы

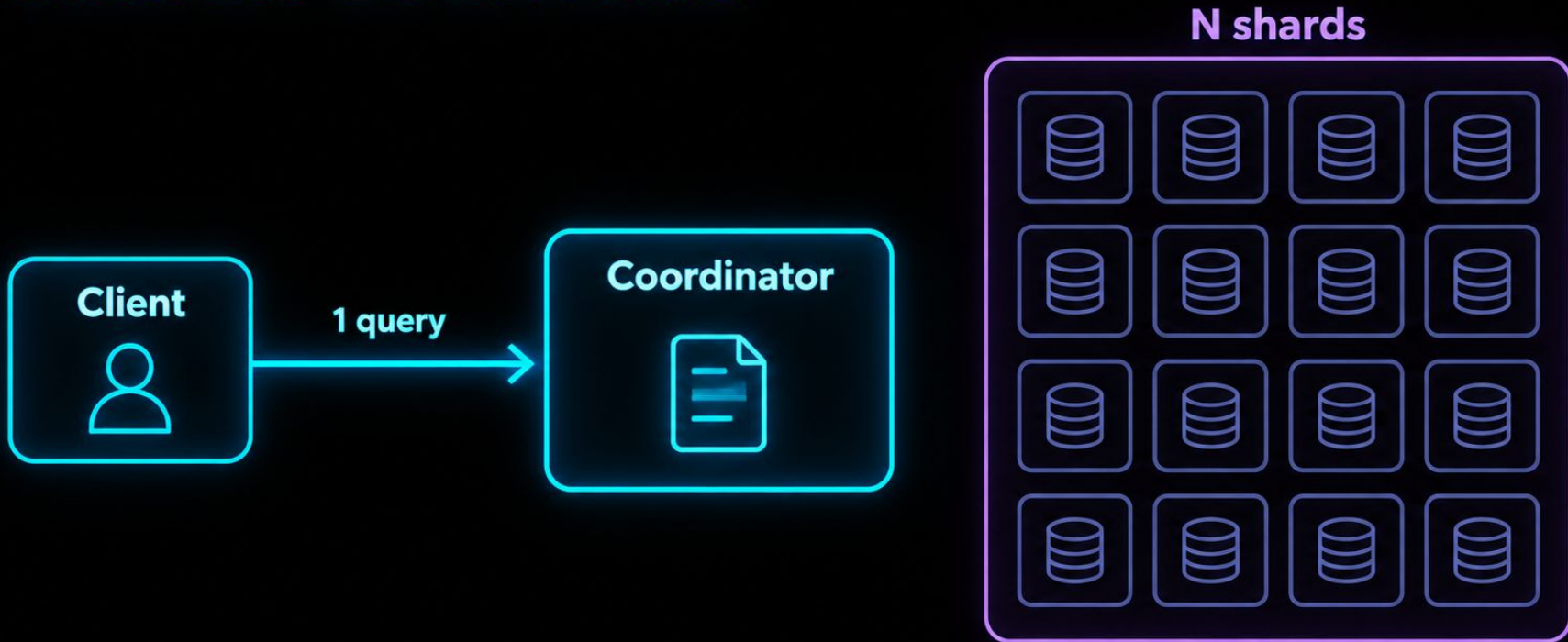
Один запрос

fan-out



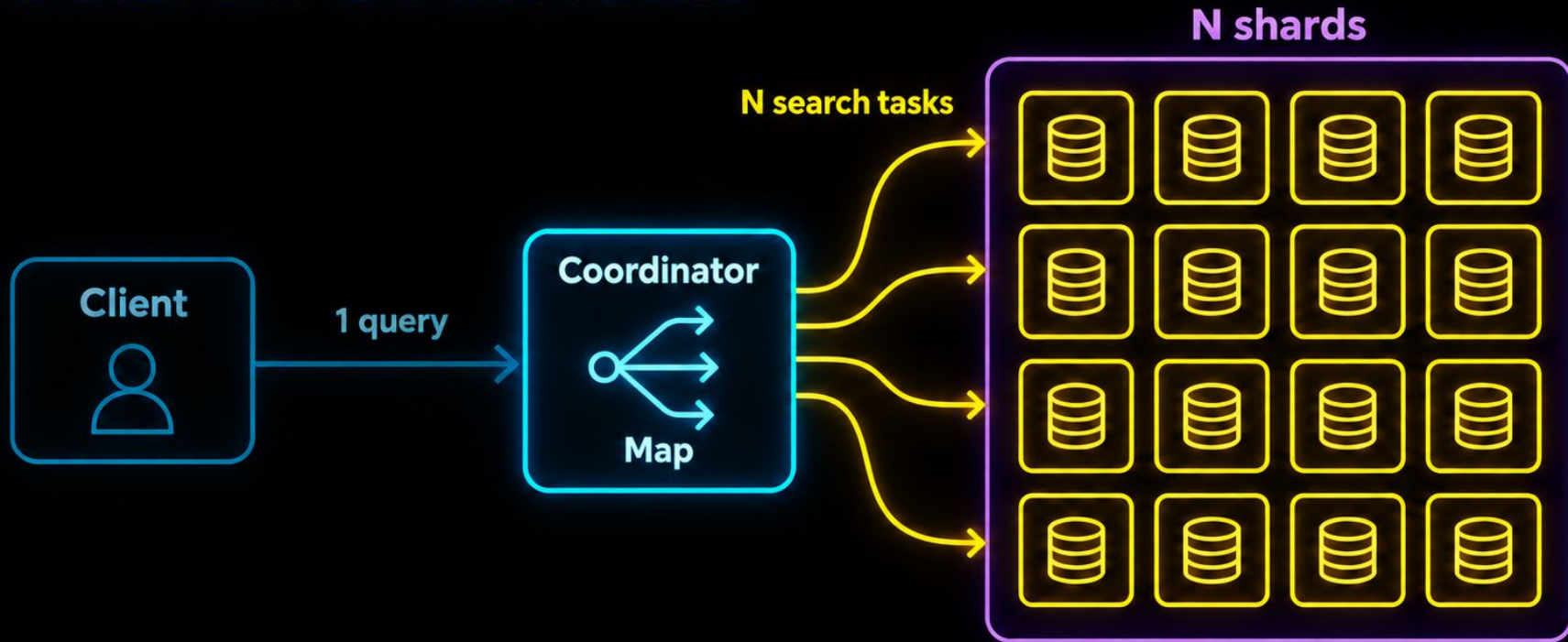
- Search overhead
больше shard-level операций
на один запрос
- Нагрузка на heap и I/O
- Operational overhead
recovery, allocation, ILM,
snapshots, segment management

Search overhead



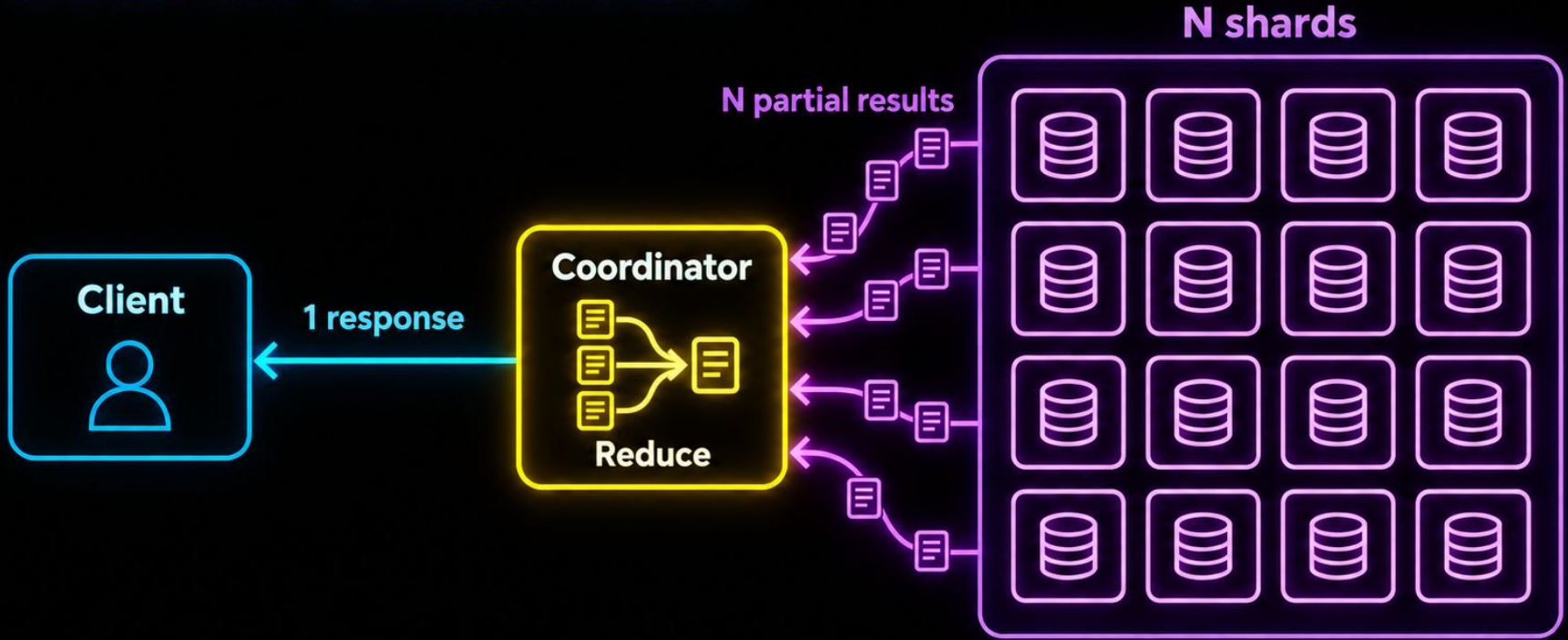
One request

Search overhead



More shards → more search tasks

Search overhead



More partial results → more reduce work

Search overhead

- У shard-level запроса есть минимальная фиксированная цена

Search overhead

- У shard-level запроса есть минимальная фиксированная цена
 - Роутинг запроса

Search overhead

- У shard-level запроса есть минимальная фиксированная цена
 - Роутинг запроса
 - Сериализация/десериализация

Search overhead

- У shard-level запроса есть минимальная фиксированная цена
 - Роутинг запроса
 - Сериализация/десериализация
 - Подготовка структур данных для поиска

Search overhead

- У shard-level запроса есть минимальная фиксированная цена
 - Роутинг запроса
 - Сериализация/десериализация
 - Подготовка структур данных для поиска
 - Частичная агрегация

Search overhead

- У shard-level запроса есть минимальная фиксированная цена
- **Tail latency**: $T_{\text{query}} \approx \max(T_{\text{shard1}} \dots T_{\text{shardN}}) + T_{\text{reduce}}$

Нагрузка на heap

Heap: 300GB, 5к шардов, 55к сегментов с данными

- Метаданные сегментов/полей:
 - TSDB doc-values entry objects - 33 GB
 - FieldInfo, FieldInfo[], and FieldInfos\$... objects - 34 GB
 - DirectMonotonicReader\$Meta - 6 GB
- Mapping:
 - Elasticsearch index.mapper.* objects - 13 GB
- RAM buffers: 15 GB

Operational overhead

Каждый шард необходимо

- Разместить/ребалансировать
- Проверить/восстановить после рестарта
- Помержить сегменты (уменьшить их кол-во)

Количество шардов

Наша конфигурация

≤ 50 GB

размер шарда

≤ 1 B docs

документов в шарде

(нужен патч)

1–5

шардов

Обычно не больше числа data nodes

2×

primary + replica
для HA

Критерии

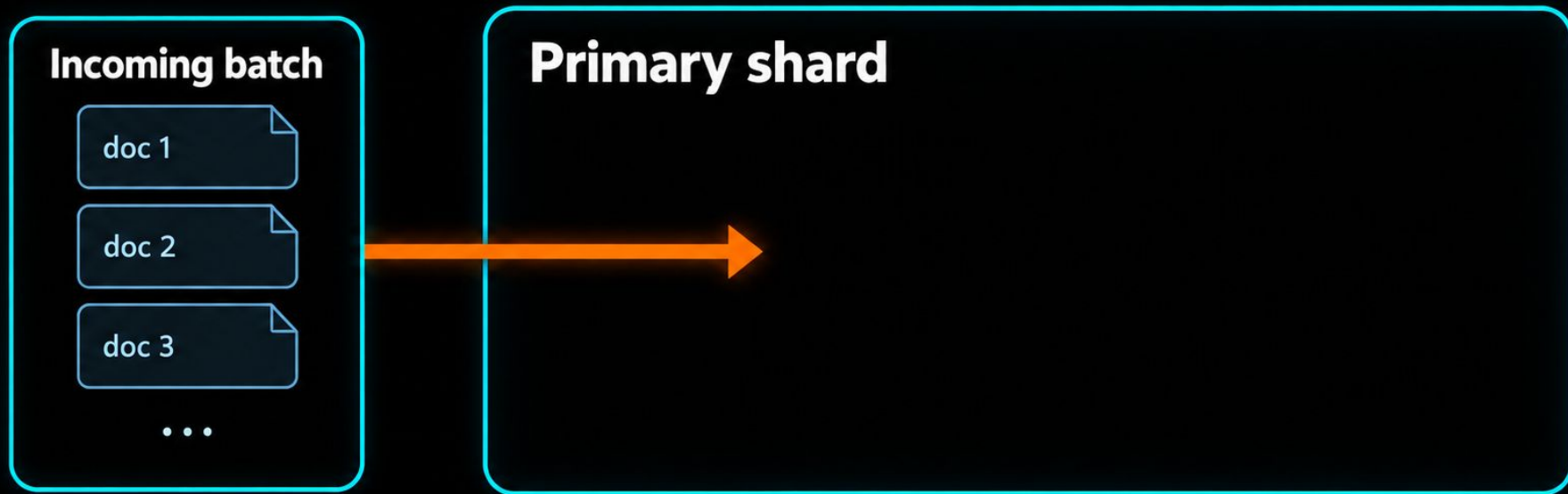
- Эффективность хранения (cost) ✓
- Шардирование (scaling out) ✓
- Репликация (HA) ✓
- Downsampling (cost)
- Надежность/эксплуатация

Прогресс 60%

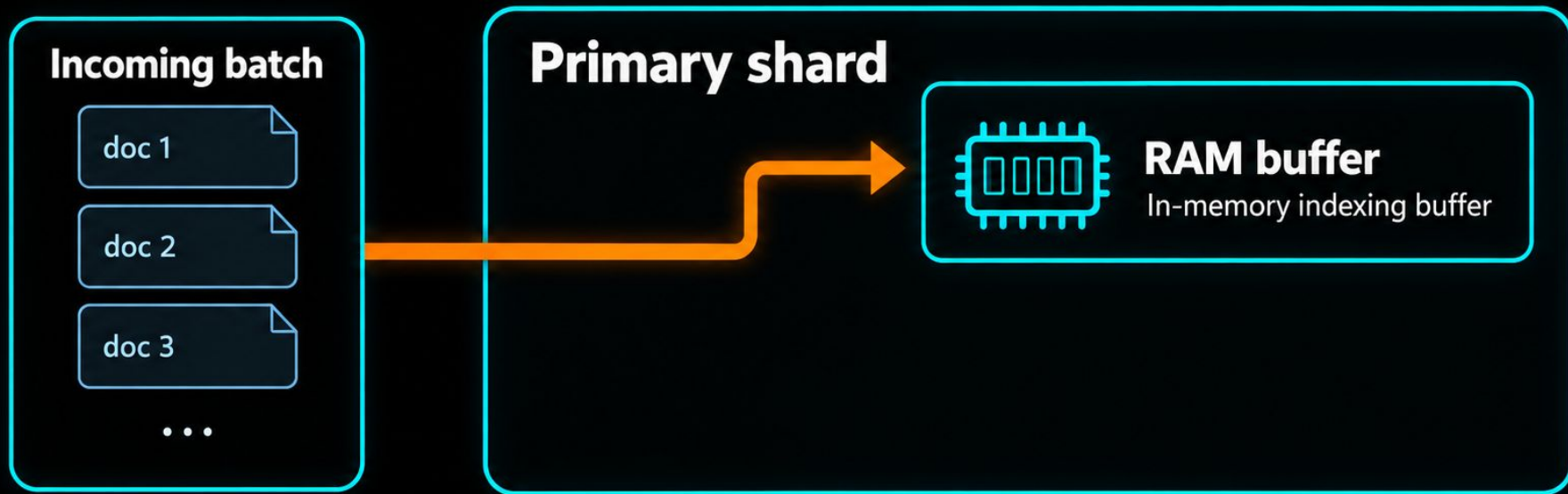
A horizontal progress bar is located at the bottom of the slide. It consists of a long thin line. The first 60% of the line is colored orange, and the remaining 40% is dark grey.

Запись **данных**

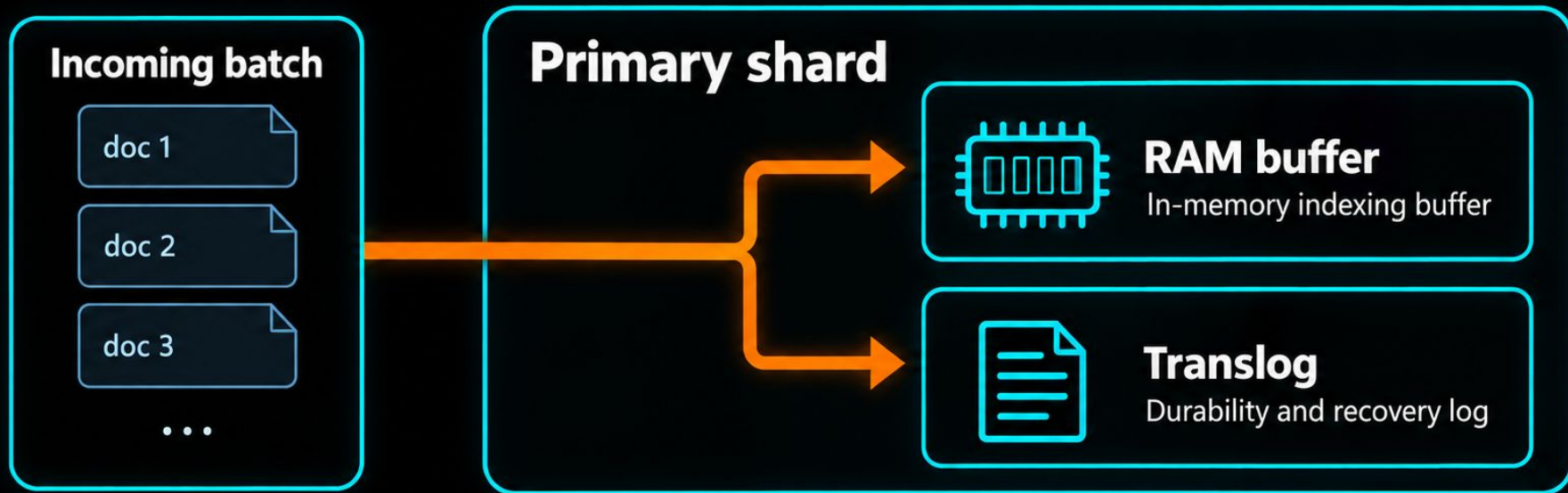
Elasticsearch **write path**



Elasticsearch **write path**

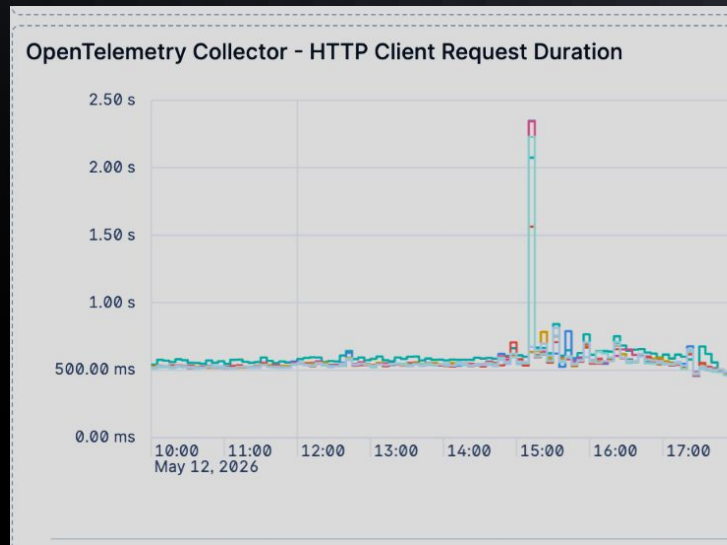


Elasticsearch write path



Запись данных

- Достигли 1 млн документов в секунду
- Latency API вставки начинает «скакать»



Запись **данных**

■ Посмотрели в thread dump:

java.lang.Thread.State: **BLOCKED** (on object monitor)

at com.fasterxml.jackson.core.util.InternCache.intern(com.fasterxml.jackson.core@2.17.2/**InternCache.java**:54)

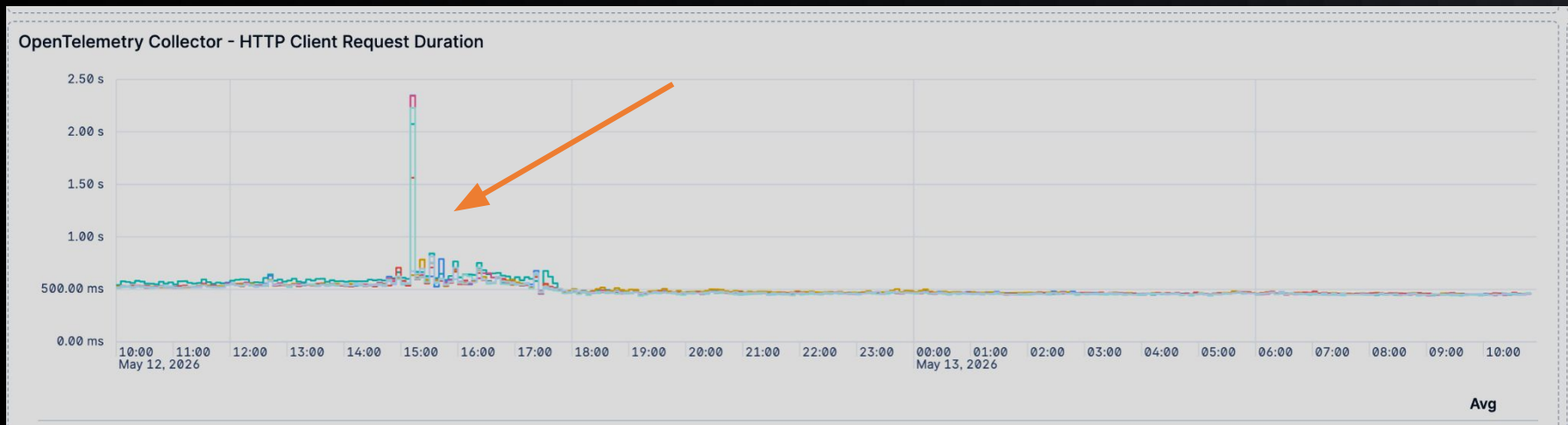
- waiting to lock <0x00007f2ba3355e10> (a java.lang.Object)

at ...ByteQuadsCanonicalizer.addName(...)

Запись **данных**

- Проблема в **Jackson 2.17.2**
- Кеш для имен полей, размер 140 элементов
- Global lock при очистке

Запись данных



- Сделал патч, но в upstream пока не взяли:

<https://github.com/elastic/elasticsearch/issues/148942>

Проблемы в дизайне записи

- У каждого шарда свой translog
 - параллельные fsync могут приводить к задержкам при записи



Проблемы в дизайне записи

- У каждого шарда свой translog
- Index и search делят одни и те же ресурсы: cпи и память

Проблемы в дизайне записи

- У каждого шарда свой translog
- Index и search делят одни и те же ресурсы: cpu и память
- Document-based репликация

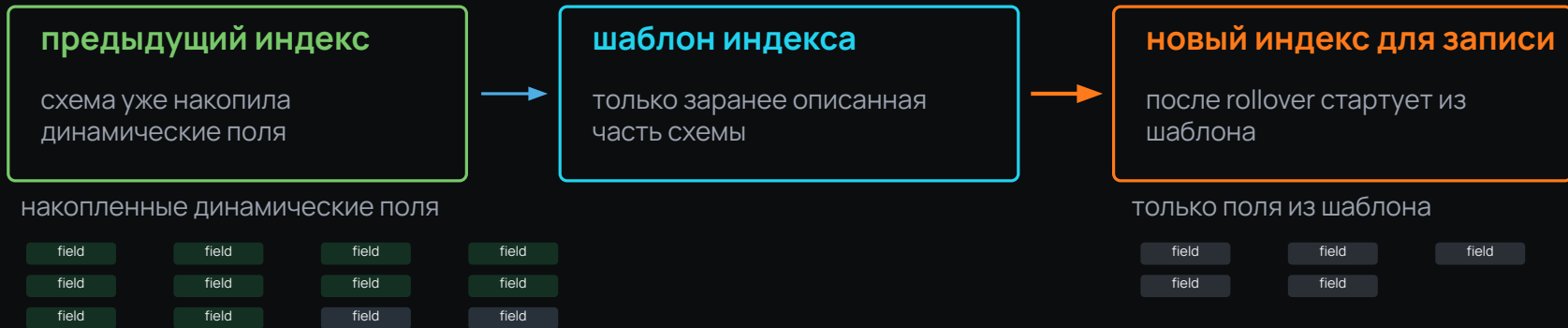
Проблема со схемой при rollover

- При переключении индексов в датастриме (rollover) заметили рост очередей на вставку

Проблема со схемой при rollover



Проблема со схемой при rollover



- Датастрим выглядит как одна сущность, но схема живет в отдельных индексах

Проблема со схемой при rollover

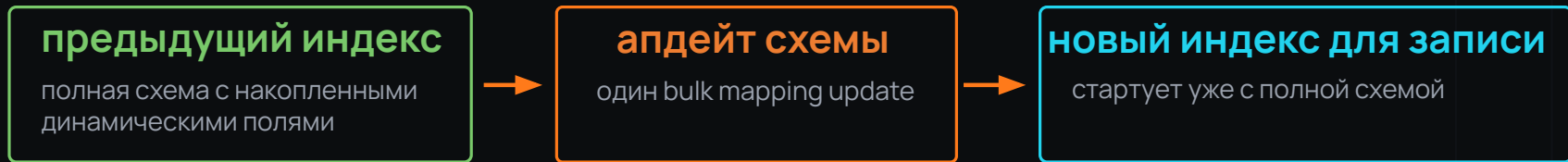
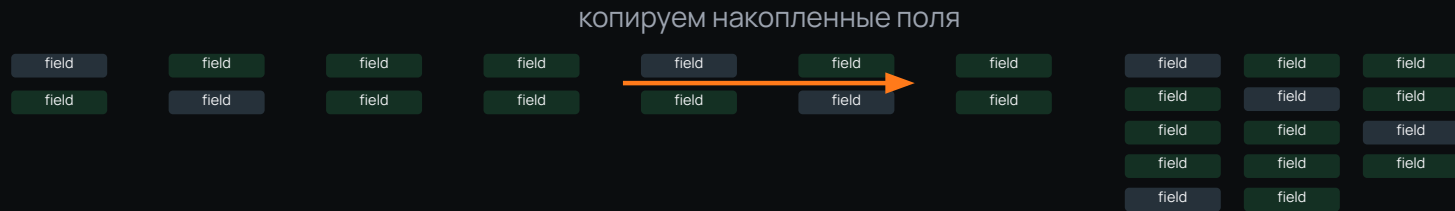


Ключевой эффект операции, которые добавляют новые поля, ждут обновления схемы

write path получает синхронную metadata-зависимость

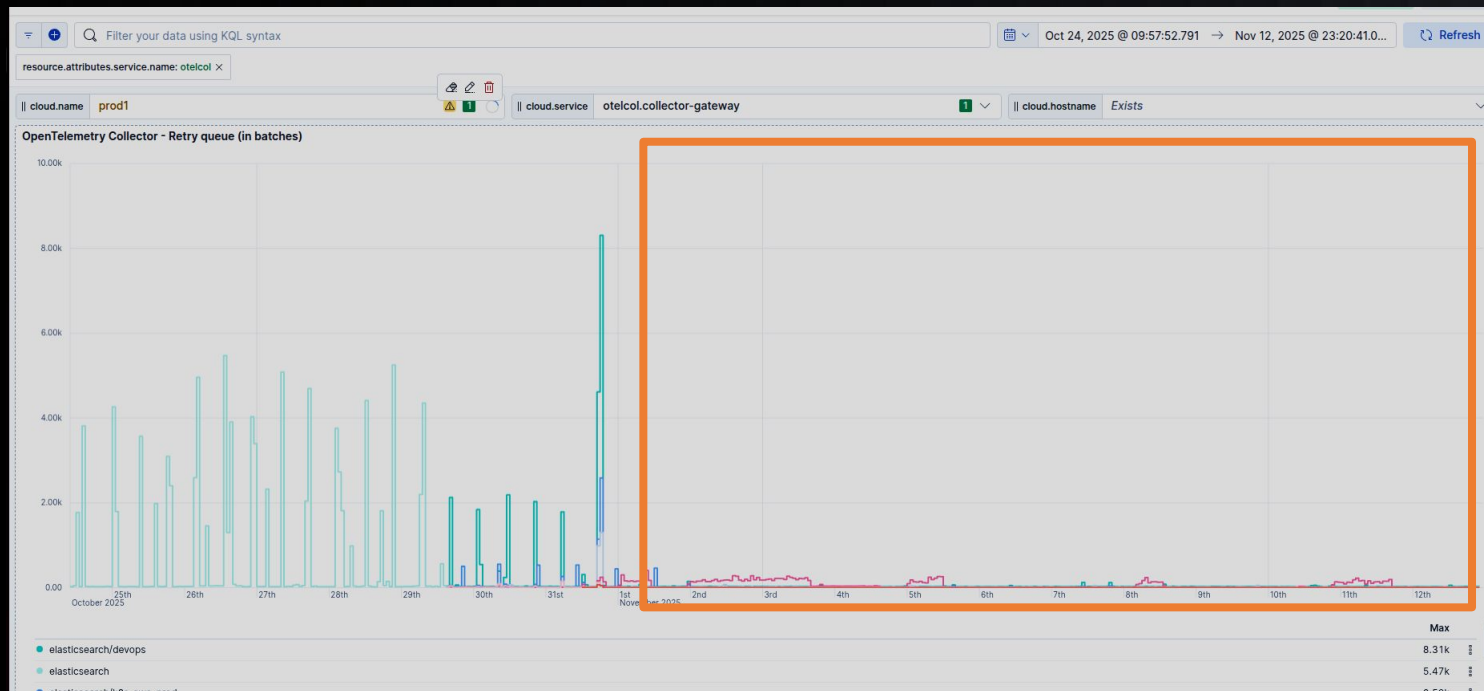
Исправление проблемы

Наследуем схему при rollover



- Берем схему предыдущего индекса
- Добавляем все поля в схему нового индекса одной операцией

Исправление проблемы



Downsampling данных

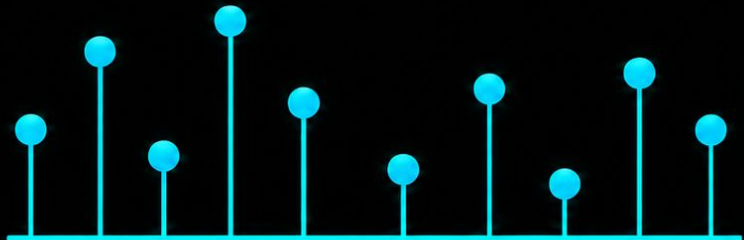
Зачем нужен **downsampling**

- Уменьшить стоимость хранения исторических данных
- Читать меньше данных при поиске и построении агрегатов

Downsampling

10 documents

Same _tsid · 5-minute bucket



1 document

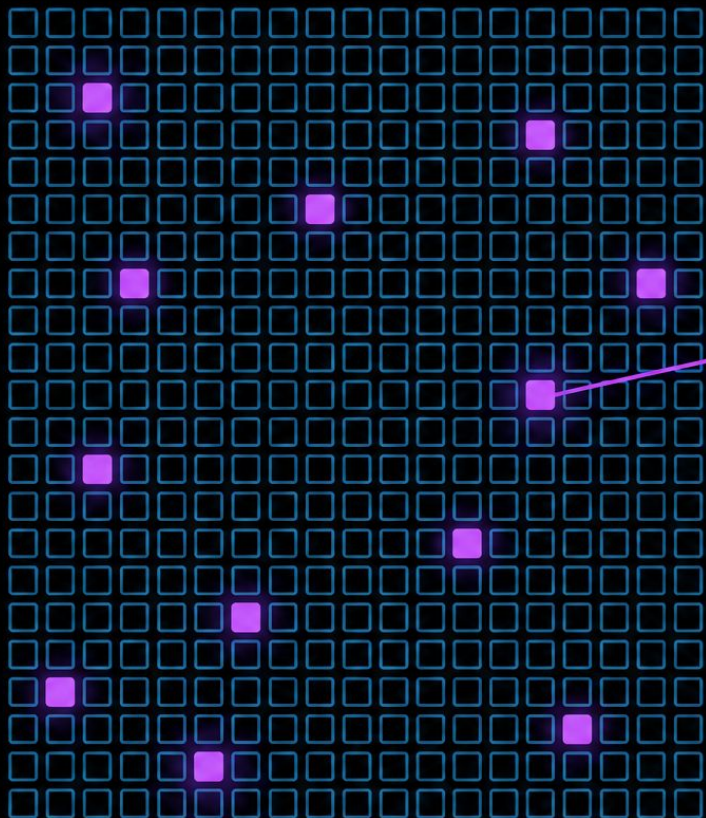
min	12
max	88
sum	356
count	10

Запустил **downsampling**

- Запустил downsampling...



One document



15K
possible fields

20-40
present

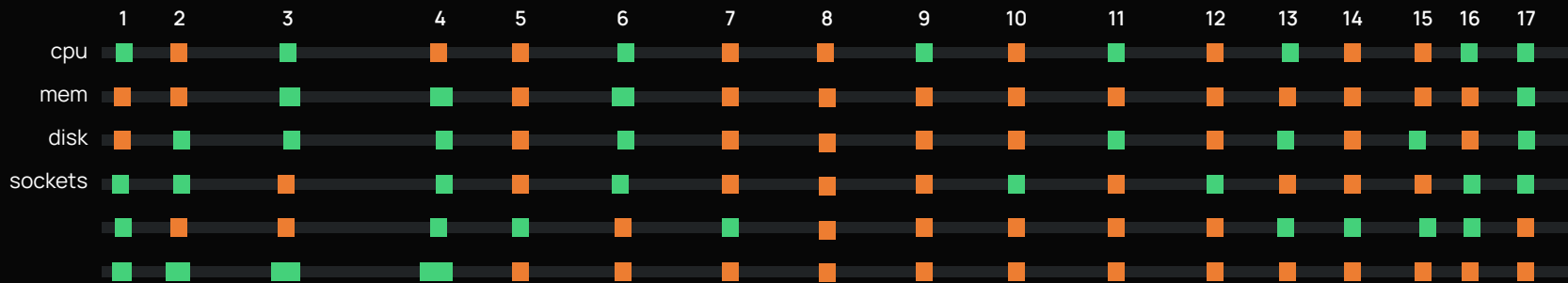
>99%
absent

Гипотеза

- Для каждого документа downsampling **проходит по всем полям** индекса, а не только по присутствующим

Наивный алгоритм **downsampling**

■ Сложность: $O(\text{total_fields} * \text{total_docs})$



■ Зеленые точки — поля, которые есть в документе

Проблема: в columnar store нет информации какие поля в каком документе

Оптимальный алгоритм **downsampling**

■ Сложность: $O(\text{avg_fields_per_doc} * \text{total_docs})$



Оптимальный алгоритм **downsampling**

End to end

НАИВНЫЙ АЛГОРИТМ ES

15.1 min



≈ 6× reduce time

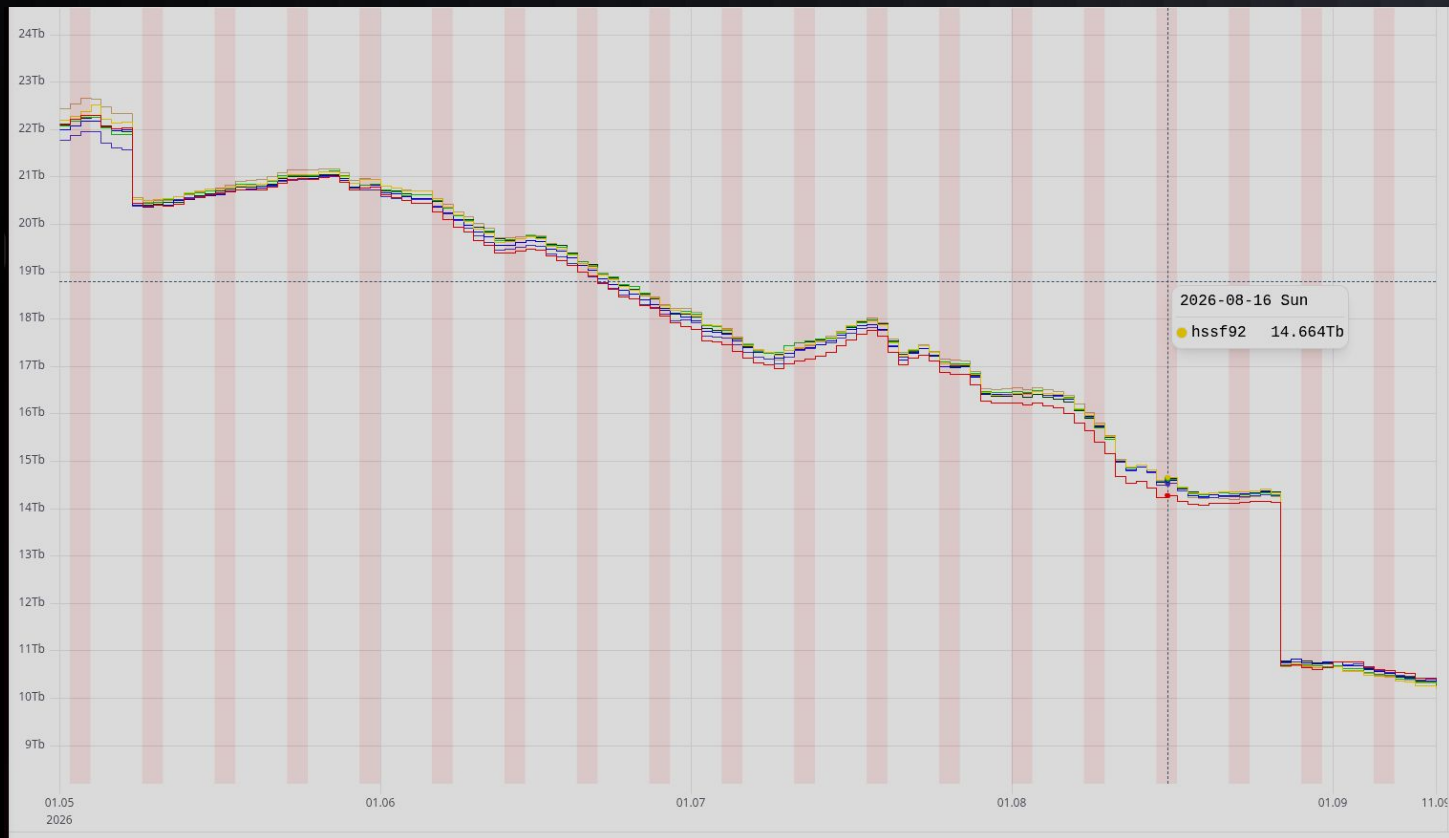
ОПТИМИЗИРОВАННЫЙ

2.4 min

- "Shard [[xxx][3]] successfully sent [26734822], received source doc [8268794], indexed downsampled doc [8268794], failed [0], took [15.1m]"
- "Shard [[xxx][3]] successfully sent [26734822], received source doc [8268794], indexed downsampled doc [8268794], failed [0], took [2.4m]"



Результат downsampling



Opensource

Merged in **ES 9.5.0**

- <https://github.com/elastic/elasticsearch/pull/144553>



sherman commented on Mar 19 • edited ▾

Contributor ...



 gmarouli approved these changes on Jun 23

View reviewed changes

gmarouli left a comment

Contributor ...

LGTM! Amazing work @sherman thank you 🚀

  2

Opensource

Contribution guidelines

01 Приносите реальную ценность и демонстрируйте ее

02 Будьте вежливы и терпеливы

03 Избегайте категоричных формулировок

04 Прислушивайтесь к ментейнерам

05 PR в open source — это не спринт, а марафон

Критерии

- Эффективность хранения (cost) ✓
- Шардирование (scaling out) ✓
- Репликация (HA) ✓
- Downsampling (cost) ✓
- Надежность/эксплуатация

Прогресс 80%

A horizontal progress bar at the bottom of the slide. It consists of a solid orange line that covers approximately 80% of the width, followed by a thin grey line for the remaining 20%.

Надежность эксплуатация

Что хорошо



Надежность

За все время не
потеряли данные



Эластичность

Расширяли/сужали
кластеры,
переезжали из ЦОДа в
ЦОД



Контроль

Можно управлять
почти всем



Поддержка

Критические баги
чинят быстро

Регулярно ломают

Performance regression in TSDS

#135340

elastic / elasticsearch
[issues/135340](#)

False-positive in boolean queries

#155936

elastic / elasticsearch
[pull/155936](#)

Documents missing from replica shards

#158212

elastic / elasticsearch
[issues/158212](#)

Советы по эксплуатации

■ Перед обновлением читать known issues

navigation_title	Known Issues
mapped_pages	https://www.elastic.co/guide/en/elasticsearch/reference/current/es-release-notes.html

Elasticsearch known issues [elasticsearch-known-issues]

Known issues are significant defects or limitations that may impact your implementation. These issues are actively being worked on and will be addressed in a future release. Review the Elasticsearch known issues to help you make informed decisions, such as upgrading to a new version.

9.5.2 [elasticsearch-9.5.2-known-issues]

- A bulk indexing operation which is being processed on the node holding the primary shard exactly when this node crosses the [indexing pressure limit](#) can process some of the bulk items on the primary shard without replicating them, causing the contents of any replicas to diverge from those of the primary. The processed items can be intermittently returned by searches while the divergence persists, but these items were not acknowledged writes and may eventually be discarded. The divergence blocks shards from trimming their [translog](#), consuming excessive disk space, and can also cause very long-running [recoveries](#) that attempt to replay all operations starting from the ones that were not replicated.

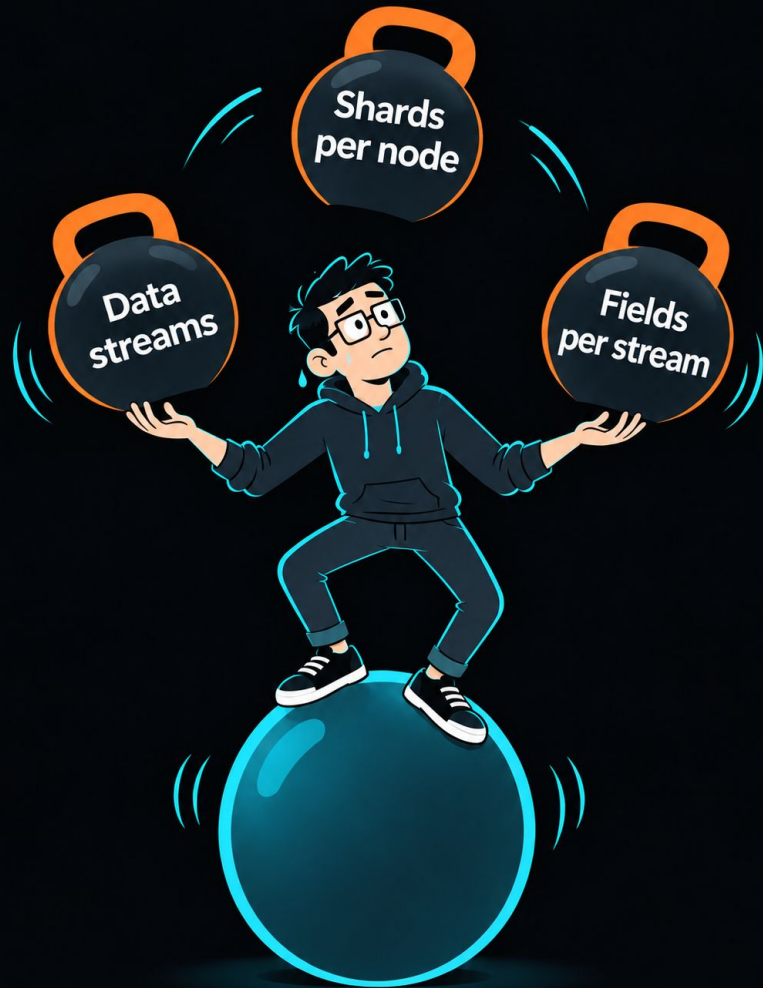
This defect was introduced in [#147151](#), released in Elasticsearch v9.5.0, and is fixed in [#158235](#), released in Elasticsearch v9.5.3. We recommend all users of earlier versions in the 9.5 series to upgrade to at least v9.5.3 as soon as possible.

Refer to [#158212](#) for further details, including information about determining whether your cluster has been affected by this issue and our recommended workarounds should any be needed.

9.5.1 [elasticsearch-9.5.1-known-issues]

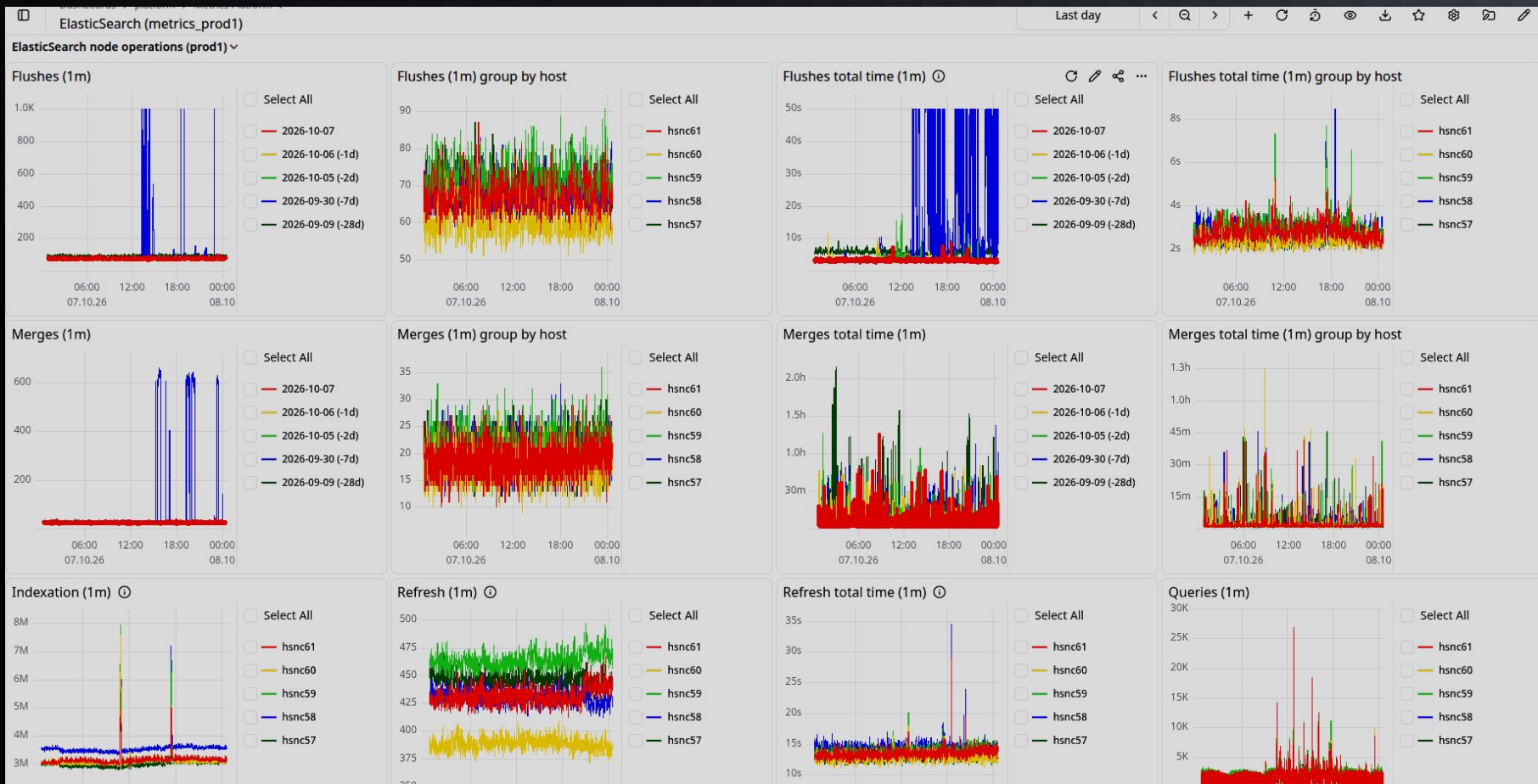
Советы по эксплуатации

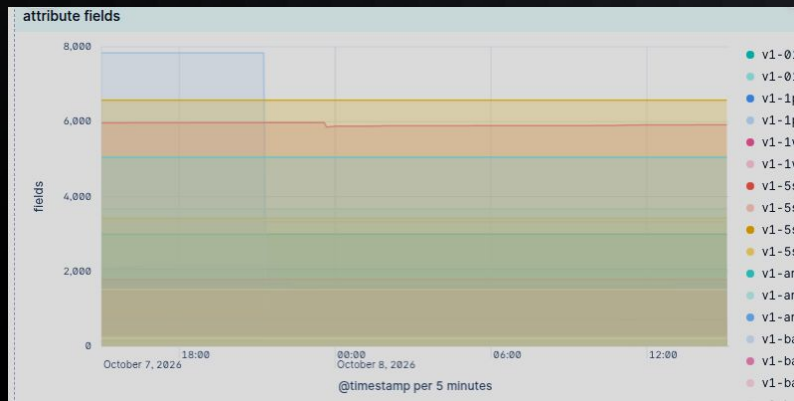
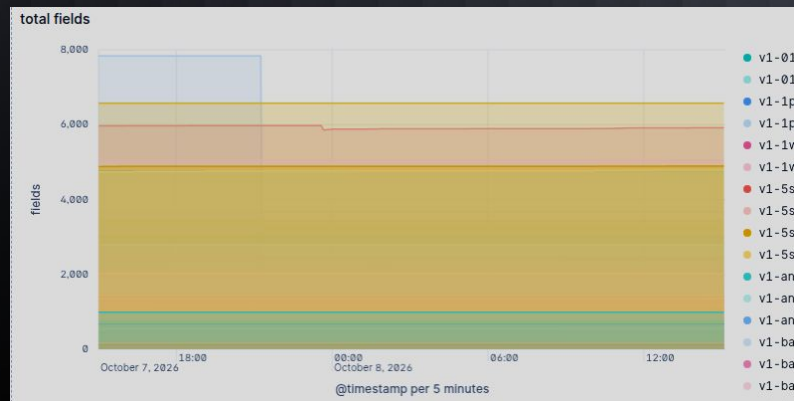
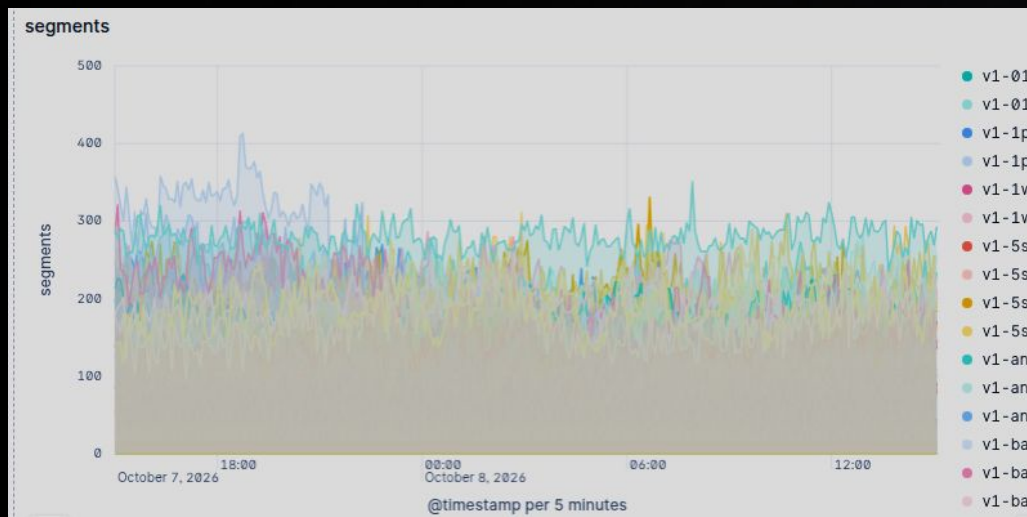
- Перед обновлением читать known issues
- Контролировать количество датастримов, шардов и полей в индексах



Советы по эксплуатации

- Перед обновлением читать known issues
- Контролировать кол-во датастримов и шардов
- Мониторить сам Elasticsearch, а не только свои данные





▼ Numeric doc-values compression

Lucene exposes numeric doc values as encoded longs, not mapping types. To keep collection time and report size bounded, diagnostics sample only the segment with the largest physical document count. Normal field and physical-byte accounting still covers every segment. Stored Lucene90 and ES95 TSDb metadata supplies exact value-stream bytes; unsupported codecs are explicitly labeled as Lucene90 estimates. A likely-double assessment is heuristic, not proof of the Elasticsearch field mapping.

Sampled segment: _1ev3 902,406,712 physical documents 3,413 numeric fields **251 poorly packed fields** 749 likely double candidates 1 modeled fields

Field filter

Assessment

Field name

All assessments ▼

☐ Exclude fields with < 1,000 values☒ Exclude fields with < 10,000 values☒ Exclude fields with < 100 KiB packed bytes

FIELD	TYPE	CODEC / MODEL	ENCODING	VALUES	BITS/VALUE	PACKED BYTES	UNIQUE	OBSERVED GCD	LUCENE90 BLOC
@timestamp	sorted numeric	ES95 TSDb	splitDelta > delta > offset > gcd > bitPack	902,406,712	3.9	419.82 MiB (440,217,447 B)	>256	15,000	not a
_primary_term	numeric	Lucene90	block delta	902,406,712	0.8	84.2 MiB (88,286,986 B)	2	1	
metrics.in_app_performance_bucket	sorted numeric	ES95 TSDb	delta > offset > gcd > bitPack	67,218,771	9.5	75.77 MiB (79,453,019 B)	>256	1	not a
metrics.service_latency_sum	sorted numeric	ES95 TSDb	alpDouble > delta > offset > gcd > bitPack	33,244,890	15.8	62.56 MiB (65,600,332 B)	>256	1	not a
metrics.external_servcies_req_time_sum	sorted numeric	ES95 TSDb	alpDouble > delta > offset > gcd > bitPack	14,077,538	17.8	29.83 MiB (31,275,193 B)	>256	1	not a
metrics.remote_service_latency_seconds_sum	sorted numeric	ES95 TSDb	alpDouble > delta > offset > gcd > bitPack	15,063,704	16.4	29.46 MiB (30,890,781 B)	>256	1	not a
metrics.service_latency_bucket	sorted numeric	ES95 TSDb	delta > offset > gcd > bitPack	56,163,557	4.1	27.39 MiB (28,718,290 B)	>256	1	not a
metrics.in_app_performance_sum	sorted numeric	ES95 TSDb	alpDouble > delta > offset > gcd > bitPack	4,158,045	50.6	25.1 MiB (26,320,673 B)	>256	1	not a
metrics.traffic	sorted numeric	ES95 TSDb	alpDouble > delta > offset > gcd > bitPack	28,138,479	7.0	23.47 MiB (24,611,965 B)	>256	1	not a
metrics.mysql_query_time_seconds_bucket	sorted numeric	ES95 TSDb	delta > offset > gcd > bitPack	29,618,522	6.0	21.28 MiB (22,316,826 B)	>256	1	not a
metrics.external_servcies_req_time_bucket	sorted numeric	ES95 TSDb	delta > offset > gcd > bitPack	34,349,772	4.2	17.07 MiB (17,898,771 B)	>256	1	not a
metrics.container.cpu.time	sorted numeric	ES95 TSDb	alpDouble > delta > offset > gcd > bitPack	2,909,273	49.0	17.01 MiB (17,834,837 B)	>256	1	not a
metrics.remote_service_latency_seconds_bucket	sorted numeric	ES95 TSDb	delta > offset > gcd > bitPack	27,965,044	4.3	14.48 MiB (15,188,937 B)	>256	1	not a

ИТОГИ

- Эффективность хранения (cost) ✓
- Шардирование (scaling out) ✓
- Репликация (HA) ✓
- Downsampling (cost) ✓
- Надежность/эксплуатация ?

Прогресс 90%



Вопросы

- gabaden@gmail.com
- tg: [sherman81](https://t.me/sherman81)

