ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет о программном проекте на тему:

Микросервис для агрегации логов

Выполнил:

студент группы БПМИ206
Енцов Семен Евгеньевич

(подпись)

18.05.2023

(дата)

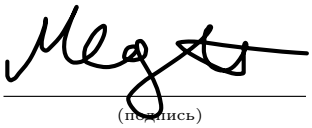
Приняли руководители проекта:

Токарев Вячеслав Сергеевич
Руководитель группы,
ООО Яндекс.Технологии

(подпись)

(дата)

Медведь Никита Юрьевич
НИУ ВШЭ



(подпись)

18.05.2023

(дата)

Москва 2023

Содержание

Аннотация

Агрегация метрик на основе журнала доступа микросервисов — полезный механизм для улучшения observability в распределённой системе. В этой работе описана реализация MVP такого агрегатора метрик в рамках инфраструктуры Яндекс Такси. Разработанный сервис предоставляет сильные гарантии на latency обработки метрик и имеет высокую степень масштабируемости.

Ключевые слова

Распределённые системы, observability, микросервисная архитектура

1 Введение

Observability в распределённых системах отражает возможности инженеров понимать, в каком состоянии находится система, по данным, которые она производит [7]. Это включает возможность собирать информацию, анализировать её и извлекать из неё полезные знания о работоспособности системы и её компонентов. Распределённые системы сложно отлаживать из-за большого числа частей, нетривиально взаимодействующих между собой, из чего вытекает необходимость в развитии observability, так как она позволяет разработчикам выявлять и устранять неполадки своевременно и эффективно. Существует множество способов улучшить observability, таких как развитие инфраструктуры для логирования и трейсинга, внедрение систем мониторинга и использование метрик для отслеживания производительности и работоспособности компонентов.

Фокус этой работы на развитии инфраструктуры для агрегации метрик по журналу доступа микросервисов. Журнал доступа — это логи, описывающие все входящие запросы к микросервису, включая характеристики запросов, такие как время обработки, адрес отправителя, url запроса и http-метод. Журнал доступа обычно ведётся балансировщиком нагрузки, таким как Nginx [8], локально на каждом узле распределённой системы. Централизованная агрегация журнала доступа позволяет получить целостное представление о работе системы, выявлять в реальном времени проблемы с производительностью и аномалии в работе компонентов и находить возможности для оптимизации.

В масштабной инфраструктуре, метрики по журналу доступа могут агрегироваться в реальном времени такими распределёнными системами, как Hadoop [9] и Spark [11]. Подобная система может получать логи запросов с каждой машины, обрабатывать их и записывать в централизованное хранилище метрик для последующей визуализации или анализа. Метрики могут быть визуализированы на графиках, что позволит в реальном времени отслеживать состояние системы, выявлять неполадки и другие аномалии в реальном времени. Метрики могут включать в себя время ответа, частоту ошибок и успешных ответов, частоту попаданий в кэш и промахов по кэшу.

В Яндекс Такси агрегацию метрик по журналу доступа выполняет сервис «дорблю». Дорблю парсит логи запросов, используя процесс наподобие демона, работающий на каждой виртуальной машине микросервисов. На этапе парсинга дорблю извлекает из логов множество метрик, используя указанную пользователем конфигурацию. Метрики, извлечённые из журналов доступа с виртуальных машин микросервисов, затем отправляются в сервис, который их агрегирует, вычисляет окончательные значения и сохраняет в другом сервисе,

хранилище метрик. Из хранилища метрики доступны инженерам для исследования и анализа.

Несмотря на то, что дорблю показал себя стабильным сервисом, будучи основным инструментом агрегации метрик по журналу доступа на протяжении нескольких лет, у него есть несколько проблем, которые было сложно решить, значительно не меняя его архитектуру. Во-первых, дорблю сложно поддерживать, так как (1) часть работы выполняется на виртуальных машинах микросервисов, из-за чего эту логику проблематично обновлять, и (2) дорблю реализован вне микросервисной архитектуры Такси на более не поддерживаемом стеке технологий. Во-вторых, модель, по которой дорблю передаёт метрики от виртуальных машин к сервису для агрегации, делает проблематичной настройку частоты сбора метрик, допустимой latency обработки логов, и ограничивает масштабируемость сервиса.

Для решения проблем дорблю, в рамках этой работы реализован альтернативный сервис для агрегации метрик по журналу доступа, решающий ту же задачу, что и оригинальный дорблю, но избегающий основных проблем, связанных с его архитектурой. В работе описана разработка MVP распределённого сервиса агрегации метрик, в реальном времени обрабатывающего поток логов микросервисов Яндекс Такси. Новая реализация дорблю вместо парсинга логов на машинах микросервисов использует распределённую очередь для доставки потока логов в микросервис, полностью выполняющий извлечение и агрегацию метрик. Предложенная система агрегации метрик обладает высокой степенью масштабируемости и имеет низкую latency с возможностью дополнительно уменьшить её за счёт потери точности метрик.

В Разделе 2 будут разобраны особенности инфраструктуры Яндекс Такси, которые повлияли на разработку, а также описаны ключевые технологии, используемые в новом сервисе. Раздел 3 посвящён дизайну сервиса: в нём в общих чертах разобран процесс потоковой обработки логов сервисом. В Разделе 4 будут описаны ключевые детали реализации нового дорблю и проблемы, с которыми пришлось столкнуться при разработке. В Разделе 5 обсуждаются альтернативные решения и работы по схожей тематике. Наконец, в Разделе 6 подводятся итоги разработки и упоминается развитие сервиса после реализации MVP. Далее в работе под «дорблю» я буду иметь в виду новый, разрабатываемый сервис.

2 Обзор окружающей инфраструктуры

При разработке сервиса для агрегации логов важно было учитывать ограничения, которые накладывала инфраструктура Яндекс Такси, и возможности, которые она предостав-

ляла. В этом разделе будут разобраны ключевые детали инфраструктуры, внутри которой был реализован сервис.

2.1 Микросервисная архитектура

В Яндекс Такси используется микросервисная архитектура — подход к разработке, в рамках которого создаются и развёртываются небольшие независимые сервисы, которые работают вместе, образуя единую систему. Эта архитектура позволяет декомпозировать монолитные приложения в небольшие слабосвязанные сервисы, которые можно разрабатывать независимо.

Каждый микросервис исполняется на кластере из нескольких виртуальных машин. На каждой виртуальной машине исполняется группа процессов, выполняющих логику микросервиса, и несколько процессов-агентов, не являющихся частью микросервиса, но обеспечивающих необходимый ему функционал. Для балансировки входящей нагрузки на виртуальную машину используется Nginx. Также Nginx отвечает за ведение журнала доступа: он пишет локально логи о входящих запросах в текстовом виде. Формат лога продемонстрирован на Иллюстрации 2.1 — одна строка такого лога описывает один входящий запрос.

```
timestamp=20220917T20:20:48  timezone=+0300
status=200 protocol=HTTP/1.1  method=GET  request=/ping
```

Рис. 2.1: Пример строки логов.

2.2 Logbroker

Инфраструктура Яндекса предоставляет полезный инструмент для потоковой обработки данных — распределённую очередь сообщений Logbroker [13]. Этот сервис позволяет передавать упорядоченный набор данных по модели Publish-Subscribe [4]. Эта модель позволяет избежать прямой коммуникации между поставщиками данных и потребителями; вместо этого поставщики сохраняют данные в распределённую очередь, а потребители читают их, «подписавшись» на требуемый им класс данных.

Одна очередь сообщений в Logbroker состоит из одной или нескольких партиций. Когда читатель подписывается на очередь, ему назначается набор партиций, из которых он читает все сообщения. Это полезное свойство, которое понадобится при реализации дорблю. Помимо этого, Logbroker предоставляет гарантию доставки `at least once` и гарантию сохранения порядка.

В Logbroker есть очередь, в которую отправляются логи запросов ко всем микросервисам Яндекс Такси. Этим занимается агент, исполняющийся на каждой виртуальной машине: он читает журнал доступа, который ведёт Nginx, группирует строки лога в сообщения и отправляет их в очередь. Эта очередь будет использоваться в дорблю для доставки логов до микросервиса.

2.3 Solomon

Конечная цель микросервиса дорблю — доставлять метрики в сервис, через который они будут доступны в удобном для анализа виде. Эту задачу решает Solomon — сервис, который хранит и визуализирует метрики, а также предоставляет инструменты для анализа метрик.

Метрики в Solomon представляются в виде временных рядов. Временной ряд — это последовательность числовых значений с указанием времени измерения каждого из них. Одна метрика характеризуется временным рядом и набором меток — пар ключ-значение. В дорблю метрики будут подсчитываться внутри последовательных временных интервалов равной длины, и в Solomon будет регулярно отправляться значение метрики внутри прошедшего временного интервала.

2.4 Наблюдения

При разработке сервиса важно было учитывать ряд наблюдений о том, как этот сервис будет использоваться. Ниже описаны ключевые наблюдения, которыми мы руководствовались.

- Дорблю должен обрабатывать поток логов около 200 МБ/сек, поступающий с порядка 1000 микросервисов. Из-за этого обработку необходимо выполнять параллельно на нескольких машинах.
- Метрики — некритичные данные. Допустимо потерять часть данных, однако важно, чтобы это не привело к значительной потере точности итоговых метрик.
- При агрегации метрик крайне важна latency — время между окончанием временного интервала и отображением значения в Solomon. Это важно для быстрого реагирования на проблемы в работе сервисов.

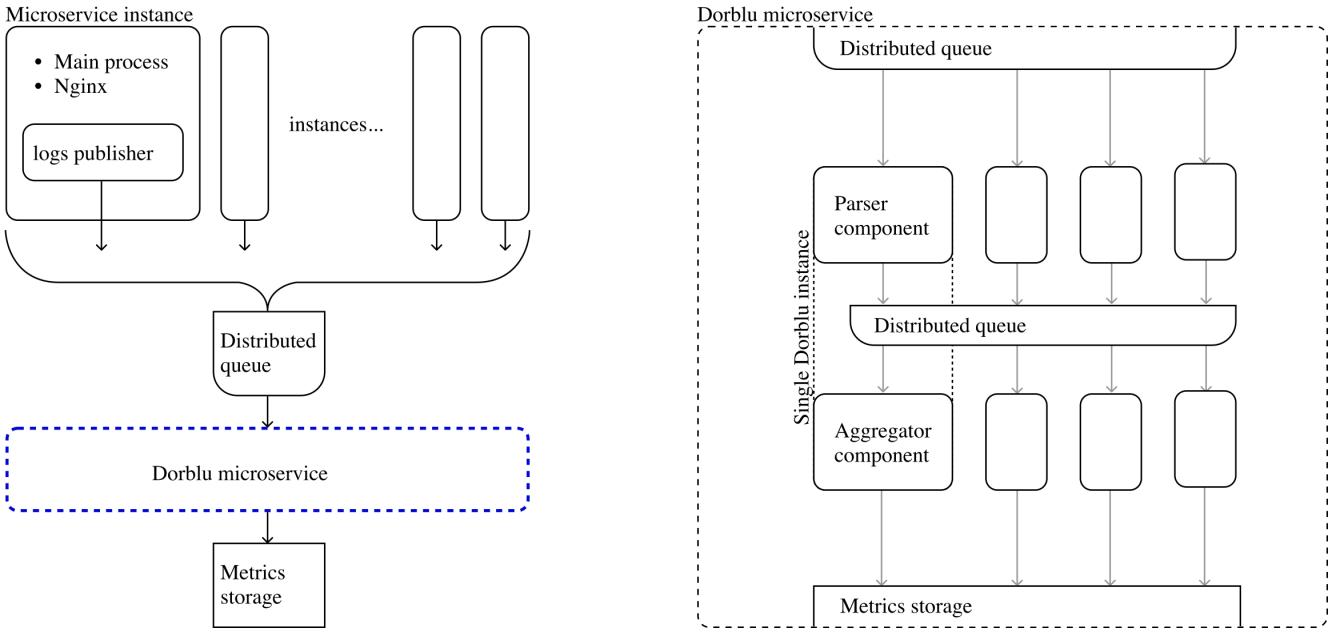


Рис. 3.1: Схема архитектуры дорблю: слева расположение микросервиса в пайплайне обработки логов, справа архитектура самого микросервиса.

3 Дизайн сервиса

Задача дорблю заключается в потоковой обработке данных: логи, непрерывно поступающие с множества машин, требуется парсить, извлекать из них метрики, согласно конфигурации, и отправлять получившиеся данные в виде временных рядов в сервис, хранящий метрики. В этом разделе будет разобран дизайн системы, выполняющей этот процесс.

В дорблю, логи обрабатываются параллельно на множестве узлов. Каждый из них читает поток логов в текстовом виде из единой очереди, парсит их, извлекает значения, необходимые для подсчёта метрик и отправляет их в отдельную распределённую очередь. Очередь промежуточных значений выполняет одну задачу — перераспределить данные так, чтобы значения из логов, принадлежащих одному микросервису, продолжили обрабатываться на одном узле дорблю. Поток промежуточных значений из очереди читают узлы дорблю, объединяют значения, принадлежащие одному временному интервалу, и, когда значения для интервала собраны, по ним вычисляются итоговые значения метрик. На Рисунке 3.1 схематично показан этот процесс.

Таким образом, в процессе обработки логов выделяется два этапа: парсинг логов и агрегация. Между этапами промежуточные данные перераспределяются между узлами. Далее будут подробнее разобраны оба этапа.

3.1 Парсинг

Парсинг логов выполняется параллельно на всех узлах дорблю. Каждый узел подписан на распределённую очередь сообщений Logbroker, по которой передаются логи всех микросервисов, которые агрегирует dorblu. Одно сообщение из очереди состоит из нескольких строк логов с одного узла микросервиса.

В конечном счёте, каждое сообщение преобразуется в список промежуточных значений. Каждый элемент в этом списке соответствует метрике, которую необходимо подсчитывать. Для этой метрики сохраняется необходимая для подсчёта информация, извлечённая из сообщения с логами. Например, для подсчёта количества запросов в секунду сохранятся количество встреченных в логах запросов, а для подсчёта среднего времени ответа сохраняется количество запросов и среднее время ответа среди них.

Полученные промежуточные значения отправляются в отдельную распределённую очередь. На этом шаге важно было гарантировать, чтобы на следующем этапе все промежуточные значения, относящиеся к одному микросервису, продолжили обрабатываться на одном узле дорблю. Это нетривиально, потому что на этапе парсинга несколько сообщений с логами, принадлежащими одному микросервису, могут быть обработаны разными узлами дорблю. Для решения проблемы используется свойство Logbroker'a, что значения из одной партиции читаются только одним узлом. После парсинга промежуточные значения отправляются в партицию Logbroker'a, которая выбирается на основании микросервиса, которому принадлежат логи. Все узлы дорблю выбирают одну партицию, из-за чего промежуточные значения одного микросервиса попадают на один узел для агрегации.

3.2 Агрегация

Агрегация метрик так же выполняется параллельно на всех узлах дорблю. Узлы читают сообщения с промежуточными значениями из распределённой очереди и сохраняют данные из них отдельно для каждого микросервиса и временного интервала, которому принадлежали логи. Через некоторое время после завершения временного интервала, по сохранённым промежуточным значениям, относящимся к нему, вычисляются итоговые метрики.

Метрики, вычисленные для данного временного интервала, доставляются в Solomon, сервис-хранилище метрик, представляющий их в виде графиков. Перед отправкой метрики сохраняются локально в формате временных рядов, который использует Solomon.

4 Детали реализации

Дорблю реализован в виде единого микросервиса на C++ при помощи асинхронного фреймворка userver [10]. Обработка логов выполняется параллельно в множестве корутин: часть из них выполняет парсинг логов, другая часть — агрегацию метрик.

4.1 Конфигурация

Дорблю обрабатывает логи порядка 1000 микросервисов, и для разных микросервисов зачастую требуется собирать разные наборы метрик. В то время как почти для всех микросервисов важно подсчитывать частоту запросов для каждой оконечной точки, многим микросервисам требуется подсчитывать более специфичные метрики. Из-за этого возникает необходимость в конфигурации собираемых метрик.

Конфигурация для дорблю хранится в базе данных; из базы данных её читают все узлы микросервиса и применяют на этапе парсинга логов. При этом возникает проблема с синхронизацией конфигурации, так как не все узлы могут обладать актуальной конфигурацией из-за задержки, вызванной сетью или тем, что часть узлов сделала запрос в базу данных позже других. В частности, проблема рассинхронизации приведёт к коллизии на этапе агрегации, если часть промежуточных значений была собрана с использованием одной версии конфигурации, а часть — с использование другой. Чтобы этого избежать, надо, чтобы во время парсинга все узлы применили одну версию конфигурации для логов из одного временного интервала, даже если часть узлов к этому моменту успела прочитать более новую версию. Чтобы это гарантировать, узлы дорблю начинают применять новую версию конфигурации не сразу после прочтения, а через фиксированное время после её записи в базу данных. Эта задержка позволяет всем узлам дорблю успеть прочитать версию конфигурации, прежде чем она вступит в силу. При достаточно большой задержке, рассинхронизация конфигурации будет происходить только в исключительных случаях, таких как отказ базы данных или разделение сети.

4.2 Latency

Одно из требований к дорблю — это как можно меньшая задержка перед доставкой метрик в Solomon. Это важно, чтобы инженеры могли быстро реагировать на неполадки в сервисах, основным показателем которых является аномалия в метриках. Для обеспечения низкой задержки, был установлен крайний срок, после которого на этапе агрегации проме-

жуточные значения за прошедший временной интервал перестают приниматься и итоговые значения метрик вычисляются по данным, полученным к этому моменту.

Задержка перед принудительным подсчётом метрик была выбрана с учётом распределения времени задержки промежуточных данных — так, чтобы не терять слишком много опоздавших данных, но и не жертвовать общей задержкой метрик ради незначительной части задержанных по какой-то причине сообщений. Таким образом, возникал компромисс между latency и точностью метрик.

Однако вне зависимости от строгости крайнего срока для промежуточных данных, оставалась часть сообщений с исключительно большой задержкой, порядка 1%. Это не много, но если потеряется значительная часть значений из одного интервала, может возникнуть нежелательная аномалия на графике. Из-за этого стояла задача аппроксимировать итоговые значения метрик по части полученных на данный момент значений. Для этого дорблю учитывает, какая часть временного интервала покрыта полученными значениями, и аппроксимирует значения на непокрытые части.

5 Результат

Реализованный микросервис выполняет поставленные перед ним задачи: обрабатывает поток логов запросов к микросервисам Яндекс Такси, извлекая из них метрики и сохраняя в сервисе для хранения метрик. Наиболее важные показатели, ожидаемые от дорблю, — точность подсчёта метрик и задержка перед доставкой в Solomon. Ниже проанализированы эти показатели.

5.1 Точность подсчёта метрик

В теории, возможно достичь идеальной точности подсчёта метрик, однако на практике метрики считаются приблизительно из-за частичной утери данных. В дорблю распределённая очередь предоставляет гарантию доставки, но из-за задержки часть данных может игнорироваться сервисом ради лучшей latency. Поэтому было важно удостовериться, что метрики подсчитываются достаточно точно. Чтобы это проверить, я сравнил итоговые метрики, производимые дорблю, и метрики, производимые старым дорблю, который использовался в Такси на момент разработки. Метрики старого дорблю тоже имеют небольшую погрешность, но они достаточно точны для пользователей, поэтому было решено принять их за эталон.

Для каждой метрики, подсчитываемой дорблю, из выборки я измерил разницу вре-

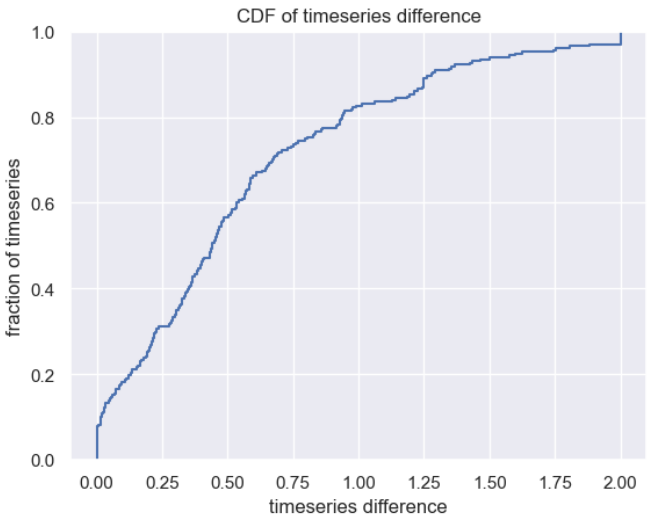


Рис. 5.1: Распределение величины различия временных рядов между старым и новым дорблю.

менных рядов, соответствующих величине метрики в течение суток. В качестве величины разницы рядов была использована следующая:

$$\frac{2\sum_{i=1}^{n}|a_i - b_i|}{\sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i}$$

где a_i, b_i — значения метрики, производимые старым и новым дорблю соответственно. Величина показывает отношение между суммарной разностью рядов к усреднённой сумме значений. Распределение этой величины приведено на Рисунке 5.1. На графике видны значительные расхождения. В основном, они вызваны различием в том, как сервисы определяют границы временных интервалов: новый дорблю проводит границы, используя время, указанное в строке лога, а в старом агент отсекает интервал по пришествии запроса от сервиса-агрегатора. Упрощая, новый и старый дорблю могли учесть строку лога в разных временных интервалах.

По результатам сравнения получилось, что 60% метрик среди всех подсчитываемых дорблю имеют разницу не больше 0.5. Это подтверждает корректность вычисления метрик новым сервисом, но и показывает наличие сильных расхождений, которыми нежелательно пренебрегать. Возможно, значения метрик различаются не только из-за разных методов выбора границ интервалов, но и из-за неточной аппроксимации метрик или из-за ошибок в работе сервиса. Для MVP это достаточно хороший результат, но для дальнейшего развития желательно исследовать причины расхождений.

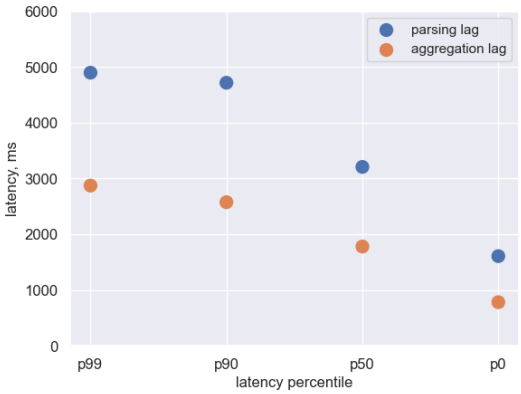


Рис. 5.2: Latency обработки логов

5.2 Latency

На Рисунке 5.2 представлен график распределения задержки метрик[1] — времени между записью строк лога и (1) отправлением промежуточных значений после их парсинга одним цветом и (2) обработкой значений на этапе агрегации — другим.

Latency на этапе агрегации получилась меньшей, чем на этапе парсинга, вероятно, потому что устаревшие промежуточные сообщения отбрасываются между парсингом и агрегацией. На графике видно, что дорблю обрабатывает поток логов с задержкой не больше 3 секунд, однако такой показатель достигается игнорированием части логов.

6 Обзор аналогов

Существует несколько технологий для агрегации метрик по журналу доступа в больших масштабах. Среди них Elastic Logstash [3], позволяющий выполнять обработку логов и, в том числе, извлечение из них метрик, и Nginx Amplify [5], работающий по принципу, схожему со старым дорблю: извлечение метрик агентами на виртуальных машинах микросервисов и их централизованная обработка. Однако ни одно из этих решений не предоставляет детальной настройки latency сбора метрик. Также их использование вместо дорблю было проблематично из-за сложности внедрения технологий в инфраструктуру Яндекс Такси и отстутствия контроля над реализацией.

Помимо специальных решений для обработки логов возможна агрегация метрик по модели map-reduce [2]. В рамках модели, обработка данных выполнятся в два этапа: на первом этапе данные преобразуются, используя произвольную функцию *map*, которая по входному значению возвращает пару ключ-значение, а на втором — пары с одинаковым ключом

[1]Эта статистика получена после доработки дорблю командой без моего участия. Вероятно, в стадии MVP задержка отличалась от этой.

объединяются, и для значений применяется функция *reduce*. Такую возможность предоставляют технологии Apache Hadoop [9] или используемый в Яндексе RealTimeMapReduce [12]. С помощью подобного сервиса можно выполнять парсинг логов в рамках первого этапа и агрегацию — в рамках второго. Однако не было найдено сервиса, выполняющего потоковую обработку данных по модели map-reduce и предоставляющего гарантии на latency, сравнимые с теми, которые требуются от .

Так же, как в дорблю, в механизмах агрегации метрик в масштабных распределённых системах зачастую требуется оптимизировать задержку между возникновением события и отображением его в системе мониторинга: например, отображением скачка RPS на графике. Дорблю обеспечивает низкую latency, обрабатывая поток логов и отправляя в хранилище только запрошенные инженерами метрики. Другой способ — сохранять логи в базе данных в оперативной памяти и извлекать метрики из них по запросу инженера. Такой подход используется в системе агрегации метрик Scuba [1]. Низкая latency обеспечивается благодаря хранению метрик в памяти, из-за чего логи, доставленные в базу данных, могут быть использованы для анализа с минимальной задержкой. Scuba предоставляет широкие возможности для анализа данных о состоянии системы за счёт высокого потребления памяти.

Наконец, перед дорблю стояла проблема перераспределения потока данных, так чтобы логи с одного микросервиса обрабатывались на одном узле-агрегаторе. В дорблю данные перераспределяются между этапами обработки через отдельную распределённую очередь. В Canopy [6], системе сбора данных о производительности компонентов, информация о производительности распределяется по шардам перед обработкой, на этапе отправки информации с машины-источника. Такой подход может привести к улучшению latency в дорблю за счёт избавления от промежуточной распределённой очереди, но он делает микросервис зависимым от реализации агента, отправляющего логи.

7 Заключение и дальнейшее развитие

В результате работы разработан MVP дорблю. Сервис успешно обрабатывает более 200 МБ логов микросервисов в секунду и агрегирует метрики на основании пользовательской конфигурации. Получившийся сервис превосходит свой устаревший аналог по нескольким параметрам: он предлагает больший контроль над latency; его удобнее поддерживать за счёт актуального стека технологий; он имеет большие возможности для масштабирования.

После реализации MVP дорблю, сервис дорабатывался, но уже не мною. На данный момент сервис, разработанный в рамках данной работы и доработанный командой после

этого, используется в Яндекс Такси наряду с его устаревшей альтернативой с планами полностью заменить устаревший сервис.

Список литературы

[1] Lior Abraham, John Allen, Oleksandr Barykin, Vinayak Borkar, Bhuwan Chopra, Ciprian Gerea, Daniel Merl, Josh Metzler, David Reiss, Subbu Subramanian и др. “Scuba: Diving into data at facebook”. В: *Proceedings of the VLDB Endowment* 6.11 (2013), с. 1057—1067.

[2] Jeffrey Dean и Sanjay Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters”. В: *Commun. ACM* 51.1 (янв. 2008), с. 107—113. ISSN: 0001-0782. DOI: 10.1145/1327452.1327492. URL: https://doi.org/10.1145/1327452.1327492.

[3] *Elastic Logstash Reference*. URL: https://www.elastic.co/guide/en/logstash/current/ (дата обр. 11.05.2023).

[4] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui и Anne-Marie Kermarrec. “The many faces of publish/subscribe”. В: *ACM computing surveys (CSUR)* 35.2 (2003), с. 114—131.

[5] *How Nginx Amplify agent works*. URL: https://amplify.nginx.com/docs/guide-how-nginx-amplify-agent-works.html (дата обр. 11.05.2023).

[6] Jonathan Kaldor, Jonathan Mace, Michał Bejda, Edison Gao, Wiktor Kuropatwa, Joe O’Neill, Kian Win Ong, Bill Schaller, Pingjia Shan, Brendan Viscomi и др. “Canopy: An end-to-end performance tracing and analysis system”. В: *Proceedings of the 26th symposium on operating systems principles*. 2017, с. 34—50.

[7] Yang-Yu Liu, Jean-Jacques Slotine и Albert-László Barabási. “Observability of complex systems”. В: *Proceedings of the National Academy of Sciences* 110.7 (2013), с. 2460—2465.

[8] Will Reese. “Nginx: The High-Performance Web Server and Reverse Proxy”. В: *Linux J.* 2008.173 (сент. 2008). ISSN: 1075-3583.

[9] Konstantin Shvachko, Hairong Kuang, Sanjay Radia и Robert Chansler. “The Hadoop Distributed File System”. В: *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)* (2010), с. 1—10.

[10] *Userver. The C++ Asynchronous Framework*. URL: https://userver.tech/ (дата обр. 11.05.2023).

[11] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker и Ion Stoica. “Spark: Cluster Computing with Working Sets”. В: *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*. HotCloud’10. Boston, MA: USENIX Association, 2010, с. 10.

[12] Вадим Никитин. *Технология Real Time MapReduce в Яндексе. Как ускорить что-то очень большое*. URL: https://habr.com/ru/companies/yandex/articles/189362/ (дата обр. 11.05.2023).

[13] Александр Озерицкий. *Logbroker: сбор и поставка больших объемов данных в Яндексе*. URL: https://habr.com/ru/companies/yandex/articles/239823/ (дата обр. 11.05.2023).