Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 21.Б15-мм

Разработка инструментария для обработки и визуализации сервисной информации

Илья Алексеевич Дудников

Отчёт по курсовой работе
в форме «Производственное задание»

Научный руководитель:
старший преподаватель кафедры системного программирования, Я. А. Кириленко

Консультант:
руководитель отдела серверных решений и СХД, ООО «КНС ГРУПП» А. А. Быков

Санкт-Петербург
2023

Оглавление

Введение

В современном мире трудно переоценить значимость данных в бизнесе. Аккуратный анализ и структуризация этих данных позволяет компаниям принимать взвешенные решения и улучшать свои продукты, получая тем самым преимущество перед своими конкурентами.

Специалисты компании YADRO также сталкиваются с необходимостью извлечения полезной информации из большого количества данных. Так, инженеры технической поддержки проводят анализ различного рода сервисной информации о состоянии систем клиентов, обеспечивая их безотказную работу. Эта информация содержится, например, в *логах* и *мгновенных снимках* (снапшотах, snapshot). Логи представляют собой журнал событий, происходящих в системе (например, ошибки во время исполнения программ или действия пользователя), который в дальнейшем может быть использован для диагностирования возникающих проблем. В этой работе нас, однако, будут интересовать снимки. Мгновенный снимок отражает состояние системы на момент его создания. Важно отметить, что снимок отличается от резервной копии системы, поскольку последняя требует полного копирования всех данных системы, что, в зависимости от размеров этих данных, может занимать большое количество времени, в отличие от снимков.

Однако информация в снимках слабо структурирована, в связи с чем их анализ становится затруднительным. Для этих целей разрабатывается инструмент *SnapShow*, позволяющий визуализировать такие данные, как характеристики накопителей, статистика по дискам (например, IOPS) и другие. Тем не менее инструмент позволяет получать лишь информацию по отдельным снимкам, но не по всем системам (или подмножеству систем).

Поэтому поступил запрос на разработку инструмента, агрегирующего данные с различных систем. Он позволит производить более широкий анализ систем и выявлять типовые и проблемные компоненты и конфигурации оборудования. В рамках этой работы предполагается разработка этого инструмента.

1 Постановка задачи

Целью работы является разработка инструмент для обработки сервисной информации с целью выявления типовых и проблемных компонентов и конфигураций оборудования. Для её выполнения были поставлены следующие задачи:

1. выделить необходимые для анализа метрики;
2. спроектировать и разработать программу для выборки данных, а также базу данных для её хранения;
3. разработать инструмент для визуализации результатов;
4. интегрировать полученные данные в имеющиеся средства аналитики.

2 Обзор

В этом разделе мы рассмотрим существующие решения для визуализации данных. Мы обсудим их преимущества и недостатки, а также актуальность в контексте этой работы.

Grafana

Grafana [2] — это стек технологий для визуализации, мониторинга и анализа данных. В него входят такие продукты, как непосредственно Grafana (для визуализации данных), Grafana Loki [3] (для агрегации логов), Grafana Pyroscope [4] (для агрегации данных, полученных в результате профилирования) и другие. Предполагается, что наилучшим образом эти решения работают в связке, то есть как единое интегрированное решение.

Процесс получения данных и их преобразование для визуализации проиллюстрирован на рис. 1. Grafana поддерживает загрузку данных из множества различных источников, например, InfluxDB [5], Elasticsearch [1], MySQL [9] и другие. Кроме того, доступно большое количество плагинов, позволяющий загружать данные из других источников, для которых нет встроенной поддержки (на момент написания этой работы доступны более 150 плагинов). К каждому источнику данных также поставляется *редактор запросов*, позволяющий настроить, как и с какой частотой запрашивать данные из источника. Затем в разделе «Transform» можно указать преобразования для данных, если они необходимы. К таким преобразования можно отнести, например, переименование полей, агрегация результатов некоторых запросов, математические вычисления и другие. Результаты преобразования также могут использоваться как входные данные для других преобразований.

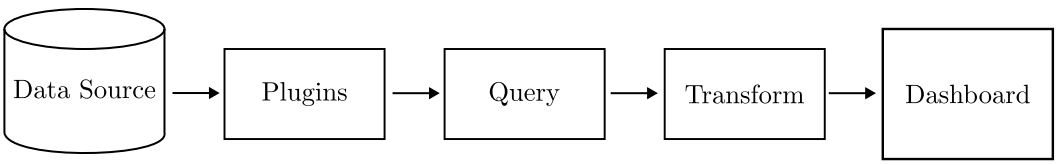

Рис. 1: Архитектура Grafana

После того, как данные получены и преобразованы, эти данные визуализируется. Grafana предоставляет широкий выбор способов визуализации, такие как круговые, столбчатые диаграммы, таблицы, гистограммы и многие другие. Пример панели, которую позволяет построить Grafana, представлен на рис. 2.

Рис. 2: Пример панели, сделанной с помощью Grafana

Кроме того, Grafana предоставляет возможность настраивать *оповещения* в случае, если происходят какие-то (заранее определенные инженером) события. Эта функциональность избавляет программиста от необходимости непрерывного наблюдения состояния системы.

Как можно заметить, основным предназначением Grafana является мониторинг системы. В рамках нашей же работы необходима визуализа-

ции редко изменяющихся данных. И хотя Grafana можно использовать и для таких целей, наличие избыточной и неиспользуемой функциональности может привести к использованию большего количества ресурсов, чем необходимо. Важно сказать, что в случае Grafana использование лишних ресурсов некритично, поскольку она менее требовательна, чем, к примеру, Kibana [6], поэтому Grafana остается предпочтительным вариантом в случае, если желаемая функциональность в разрабатываемом инструменте потребует использование более сложных решений.

Наконец, Grafana оптимизирована для работы с *временными рядами* (time series). Нас же интересует лишь последняя доступная информация о системах, а не их изменение во времени.

Kibana

Kibana [6] — это продукт из стека ELK (Elasticsearch [1], Logstash [8], Kibana), предназначенный для визуализации данных. Аналогично рассмотренной ранее Grafana, в Kibana доступны различные способы загрузки данных. Получение данных из источников осуществляется с помощью раздела «Discover», а для фильтрации данных используется *Kibana Query Language* [7]. Одним из ключевых отличий Kibana от Grafana является отсутствие встроенной функциональности для преобразования данных. Эти преобразования можно осуществлять в других продуктах из стека (Elasticsearch и Logstash), что, однако затрудняет использование Kibana для визуализации в изоляции от других продуктов. Кроме того, она лучше всего приспособлена для работы с временными рядами, хотя не ограничивается только ими. Пример панели, построенной с помощью Kibana, представлен на рис. 3.

Также ELK предоставляет встроенные функции для использования машинного обучения. Например, функция *выявления аномалий* моделирует нормальное поведение системы, основываясь на данных за прошедшее время, что позволяет ей определять моменты времени, когда возникают непредвиденные и/или вызывающие ошибки события. Возможно также настроить оповещения о нахождении аномалий. К

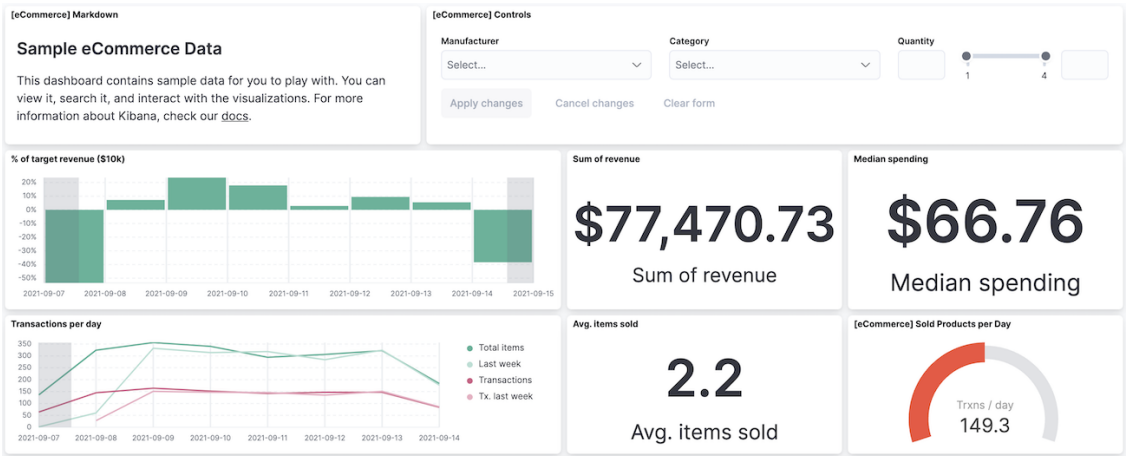

Рис. 3: Панель в Kibana

сожалению, эти функции платные, что на текущий момент не позволяет воспользоваться их преимуществами. К тому же, для использования машинного обучения в Kibana необходим Elasticsearch.

В общем случае, вероятно, нет проблемы в использовании всего стека целиком, или хотя бы интегрированного решения в виде Elasticsearch и Kibana. Для наших текущих требований, однако, накладные расходы на ресурсы, необходимые для интегрирования Elasticsearch, не оправданы. Впрочем, в рамках этой работы разработка инструмента выполняется с предположением, что может возникнуть потребность в использовании и таких инструментов, как Kibana или описанная ранее Grafana, даже если на текущем этапе такой необходимости нет.

Plotly и Plotly Dash

Plotly [10] — это набор библиотек для построения интерактивных графиков. Самой популярной и распространенной из них является библиотека для языка Python, но доступны также и решения для других языков, таких как, например, R, JavaScript, F# и прочих.

Поскольку Plotly выполняет лишь построение графиков, программист не ограничен в том, что использовать в качестве источника данных и как эти данные трансформировать, что является безусловным преимуществом в сравнении с описанными выше решениями. Ограничение возникает лишь из-за количества и разнообразия доступных видов гра-

фиков, где Plotly не уступает Kibana и Grafana.

Plotly также предоставляет фреймворк *Plotly Dash* [11], абстрагирующий написание HTML разметки и позволяющий построить панель визуализации целиком на Python. Это, в том числе, позволяет делать панели более интерактивными, чем возможно в других решениях. Поскольку разметка страницы задается динамически, появляется возможность отслеживать различные события и реагировать на них изменениям графика или даже нескольких графиков. К примеру, такими событиями могут быть нажатия кнопки мыши или наведение курсора на элементы графиков. Ещё одной возможностью является кросс-фильтрация, то есть фильтрация одновременно нескольких графиков, если они к примеру, отражают связанные данные (или одинаковые данные, но с разных перспектив).

Список литературы

[1] Elasticsearch. — URL: https://www.elastic.co/elasticsearch.

[2] Grafana. — URL: https://grafana.com/.

[3] Grafana Loki. — URL: https://grafana.com/oss/loki/.

[4] Grafana Pyroscope. — URL: https://grafana.com/oss/pyroscope/.

[5] InfluxDB. — URL: https://www.influxdata.com/.

[6] Kibana. — URL: https://www.elastic.co/kibana.

[7] Kibana Query Language. — URL: https://www.elastic.co/guide/en/kibana/current/kuery-query.html.

[8] Logstash. — URL: https://www.elastic.co/logstash.

[9] MySQL. — URL: https://www.mysql.com/.

[10] Plotly. — URL: https://plotly.com/.

[11] Plotly Dash. — URL: https://dash.plotly.com/.