УДК 004.052.2

МЕТОДЫ И ПОДХОДЫ К ЛОГИРОВАНИЮ И ЗАМЕРАМ ПРОИЗВОДИТЕЛЬНОСТИ ПРИ РАЗРАБОТКЕ ПРИЛОЖЕНИЯ «АВТОМАТИЗИРОВАННАЯ СИСТЕМА КОММУНИКАЦИЙ СТУДЕНТОВ, АДМИНИСТРАЦИИ И ПОТЕНЦИАЛЬНЫХ РАБОТОДАТЕЛЕЙ»

В.И. БАКЛАН
(Представлено: канд. техн. наук, доц. И.Б. БУРАЧЁНОК)

Рассмотрены способы и походы к применению систем логирования на примере разработанного приложения «Автоматизированная система коммуникаций студентов, администрации и потенциальных работодателей». Показаны примеры использования утилит и систем для работы с собранными данными лога.

Надежное программное обеспечение можно создать с помощью тщательной разработки архитектуры и выявления ошибок в компонентах, которые больше всего оказывают влияние на надежность системы. Эти компоненты определяются как наиболее часто используемые или архитектурно связанные с множеством других компонентов, влияя, таким образом, на их надежность. Если в работе сервера, компьютера или программного обеспечения возникла неизвестная ошибка, в первую очередь смотрят логи (текстовые файлы с зафиксированной хронологией событий, ошибок и пр.).

Логированием называют запись логов. Оно позволяет мониторить состояние системы в режиме реального времени, понимать, что происходило и при каких обстоятельствах. Во время возникновения исключительной ситуации поиск корня проблемы является критичным, так как клиенты, у которых часто происходит отказ системы, не будут готовы сотрудничать в будущем, что приведет к финансовым проблемам. Кроме этого, без логов сложно понять причину возникновения ошибки, если она появляется исключительно при совпадении нескольких условий одновременно. Чтобы облегчить задачу администраторам и программистам, в лог записывается информация не только об ошибках, но и о причинах их возникновения.

Кроме сбора логов необходимо постоянно собирать метрики о работоспособности системы. Примером метрик может быть количество ошибок за единицу времени, количество обработанных запросов в систему, среднее время обработки запроса и так далее. Все эти метрики помогают определить системные аномалии, которые в последствии приводят к некорректной работе системы. Аномальные ситуации обычно хорошо видны на графиках, что позволяет людям, которые не знают код приложения, определить наличие проблемы и попытаться сопоставить аномальные метрики с логами. Это позволяет разработчикам делегировать мониторинг системы на других участников.

Существуют разные уровни и разные подробности логирования. В корневом сервисе данного приложения использовался log4j, которые позволяет конфигурировать следующие уровни логирования:

– ALL – логирование всех событий. Данный уровень существенно замедляет приложения, так как он выводит все логи в поток вывода, что является достаточно сложной и долгой операцией.

– DEBUG – логирование всех событий при отладке. Очень редко включается в продакшене, обычно используется для отладки во время разработки или во время дебагга.

– INFO – логирование ошибок, предупреждений и информационных сообщений. Этот уровень логирования идеально подходит для продакшена, так как он включается в себя основные информационные сообщения, а также позволяет быстро идентифицировать ошибки.

– WARN – логирование ошибок и предупреждений.

– ERROR – логирование только ошибок.

– FATAL –логирование только ошибок, приводящих к прекращению работы компонента, для которого ведется логирование.

– OFF – логирование отключено.

Для работы с логами и поиском информации в огромных текстовых данных используют специализированные инструменты. Обычно эти инструменты представляют собой движок полнотекстового поиска с собственным языком запросов. Кроме этого, для метрик используются свои инструменты, которые позволяют визуализировать собранные метрики.

Логи всех информационных систем, подключенных к разработанному приложению, хранятся в распределенном хранилище на базе решения ELK (Elasticsearch, Logstash и Kibana). Механизм сбора логов выглядит следующим образом: Logstash собирает логи и переносит их в хранилище, Elasticsearch помогает найти нужные строки в этих логах, а Kibana визуализирует их. Все три компонента разработаны на основе открытого кода, благодаря чему их можно всегда модифицировать под потребности компании.

Logstash – приложение для работы с большими объемами данных, собирает информацию из разных источников и переводит ее в удобный формат.

Elasticsearch – система для поиска информации. Помогает быстро найти нужные строки в файлах хранения.

Kibana – плагин визуализации данных и аналитики в Elasticsearch. Помогает обрабатывать информацию, находить в ней закономерности и слабые места.

В разработанном приложении логи писались в каждом микросервисе, что позволяет просматривать целостные логи всей системы, отсортированные по времени. Хранение логов, согласно изначальной архитектуры, производится в системе ELK, что позволяет быстро генерировать выборки и находить проблемные места приложения.

Проблемой сбора логов в веб-приложениях является их конкурентная обработка запросов. То есть, веб-сервер может одновременно принимать и обрабатывать несколько запросов в разных потоках. Из-за этого возникает проблема, что логи пишутся асинхронно из разных потоков и в случае дебагга необходимо их как-то дифференцировать.

Для решения этой проблемы есть несколько возможных решений.

Например, можно логировать ID-поток, но с этим походом тоже имеется своя проблема. Во-первых, один запрос может создавать побочные потоки, во-вторых, в случае с микросервисной архитектурой, запрос обрабатывается разными сервисами на разных машинах, так что ID-потока будет разный. Решением этих проблем является добавление в лог случайной строки, которая называется trace ID. Trace ID генерируется случайно при первом обращении к системе и добавляется к каждому сообщению, которое выводит информацию об этом конкретном запросе. Конечно, есть ряд задач, которые необходимо решить при использовании trace ID. Например, как передавать trace ID между микросервисами, как передавать trace ID между потоками, как генерировать уникальный trace ID.

Но, решение всех этих проблем и интеграция trace ID в свое приложение позволит тратить заметно меньше времени на поиск проблемы и отладку системы, если что-то идет не по плану как в продакшене, так и во время тестирования. Если затронуть тему логирования во время тестирования приложения – в данной ситуации имеются свои проблемы. Обычно система пишет очень много логов, и эти логи обрабатываются сторонними системами (например, ELK). Для работы этих систем необходимо наличие отдельных серверов, и, порой, поддержка подобных систем стоит значительного дорого. В случае с продакшеном – это необходимые затраты. Однако если для тестовых или локальных окружений хочется сэкономить, можно всегда воспользоваться стандартным потоком вывода (stdout) и искать проблемы локально. Это не должно быть проблемой, так как для тестирования или разработки, как правило, обрабатывается только один поток одновременно и поиск проблем не будет невыполнимой задачей.

Что касается сбора метрик при локальной разработке и тестировании, надо понимать, что не стоит отправлять метрики в систему, которая работает в продакшене, так как в этом случае показания будут искажены. В большинстве случаев сбор метрик при разработке или тестировании не нужен, так что нужно предусмотреть возможность отключения метрик.

Отдельно, стоит рассмотреть мониторинг запросов к базе данных. Обычно узким местом веб-приложения являются неэффективные запросы. Иногда написание неэффективного запроса является необходимостью, иногда случайностью, так что хорошей практикой является создание отдельного dashboard, на котором предусмотрен вывод топ самых медленных запросов к базе данных в системе. В дальнейшем на основании этого топа запросов предполагается выстраивать стратегию оптимизации системы.

ЛИТЕРАТУРА

1. Learn ELK Stack // Saurabh Chhajed. – Packt Publishing, 2015. – 390 с.
2. Logging in Action // Phil Wilkins. – Hinning, Inc, 2020. – 290 с.