УДК 004.75

ПОДХОДЫ К ОРГАНИЗАЦИИ ТРАССИРОВКИ ЗАПРОСОВ В РАСПРЕДЕЛЁННЫХ СИСТЕМАХ С МИКРОСЕРВИСНОЙ АРХИТЕКТУРОЙ

Гагарин В. Ю. (Университет ИТМО), **Вагнер А. В.** (Bidease),
Научный руководитель – кандидат технических наук, доцент Тропченко А. А.
(Университет ИТМО)

Последнее время в архитектуре программного обеспечения существует тенденция на смещение акцента в сторону систем с микросервисной архитектурой. Растёт в геометрической прогрессии как сложность, так и количество пользователей, которые этими системами пользуются. В таких системах бывает сложно понять, по какой именно причине запрос завершился ошибкой, или почему время его выполнения сильно превысило допустимые границы. Рассмотрим основные подходы к организации трассировки запросов в распределённых системах с микросервисной архитектурой, как вариант решения проблемы неконтролируемой сложности мониторинга подобных систем.

Введение. При обычном логировании отдельных операций каждого микросервиса в лог-файл сложно отследить весь стек вызовов. Браузерный запрос порождает сотни вложенных запросов в разные сервисы. Непонятно, что привело к вызову тех или иных операций, сложно отследить последовательность действий. Если в системе появилось узкое место в производительности, то при таком подходе достаточно много времени тратится на мониторинг сервисов и разбор логов. Существуют подходы трассировки запросов, когда каждый вызов в рамках запроса, именуемый span'ом, фиксируется в дереве исполнения запроса для последующего отложенного анализа. Рассмотрим основные из них.

Основная часть. Собственная разработка проводится на базе высоконагруженной платформы Torrow. Платформа имеет микросервисную архитектуру. В качестве основного средства разработки используется фреймворк .NET Core. Микросервисы платформы разделены на логические слои по своим функциональным возможностям и уровню доступа к данным. В качестве протокола межсервисного взаимодействия используется протокол GRPC – система удалённого вызова процедур от компании Google. Цель состоит в написании библиотеки GrpcInterceptor, которая будет скрывать реализацию ZipkinTracing и JaegerTracing за интерфейсом ITracing. В таком случае при конфигурации и старте микросервиса мы сможем настраивать систему трассировки с помощью конфигурационного файла. Существует общий стандарт OpenTracing, который описывает, как данные должны агрегироваться, не привязываясь к конкретной реализации. Zipkin и Jaeger – имплементация OpenTracing. Основные сущности выглядят следующим образом:

- Span – конкретный вызов RPC;
- Trace – набор вызовов RPC, которые произошли в рамках одного запроса;
- TraceId – идентификатор запроса;
- SpanId – идентификатор отдельного вызова RPC в рамках запроса;
- ParentSpanId – идентификатор родительского вызова RPC, который породил текущий вызов;

Выводы. Рассмотрены основные подходы к организации трассировки запросов в распределённых системах с микросервисной архитектурой. Разработан программный модуль интеграции микросервиса в распределённую систему трассировки запросов и карту зависимостей на базе существующих решений от Zipkin и Jaeger. Модуль подключается как внешняя библиотека. Дана оценка эффективности разработанного решения. Подсистема

внедрена в промышленную версию разрабатываемой высоконагруженной платформы Torrow. Планируется дальнейшее апробирование результатов в научно-образовательном журнале.

Список использованных источников:

1. OpenTracing guide JAVA, 2023, [Электронный доступ], URL: https://opentracing.io/guides/java/
2. Evolving Distributed Tracing at Uber Engineering, 2023, [Электронный доступ], URL: https://eng.uber.com/distributed-tracing/
3. Running Jaeger Agent on bare metal, 2023, [Электронный доступ], URL: https://medium.com/jaegertracing/running-jaeger-agent-on-bare-metal-d1fc47d31fab