DOI 10.37882/2223–2966.2020.04.15

ИДЕМПОТЕНТНОЕ API В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ. АНАЛИЗ И ПРЕДЛОЖЕНИЯ ИХ РАЗВИТИЯ

IDEMPOTENT API IN DISTRIBUTED SYSTEMS. ANALYSIS AND SUGGESTIONS FOR THEIR DEVELOPMENT

A. Elizov

Summary. A comparative analysis of the idempotent API of some distributed systems is given. The problem of possible violation of data consistency when using the GET method in the HTTP protocol when requesting resources with a strict accounting of the number of hits is shown. A proposal was developed to take this feature into account when developing an idempotent API.

Keywords: idempotent API, distributed systems, idempotency key, data consistency.

Елизов Александр Иванович
Аспирант, Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)
sashacrane@yandex.ru

Аннотация. Приводится сравнительный анализ идемпотентного API некоторых распределенных систем. Показана проблема возможного нарушения согласованности данных при использовании GET-метода в протоколе HTTP при запросах ресурсов со строгим учетом количества обращений. Формулируется предложение по учету указанной особенности при проектировании идемпотентного API.

Ключевые слова: идемпотентное API, распределенная система, Idempotency Key, согласованность данных.

Введение

Приобретение товаров на площадке, обмен сообщениями в социальных сетях, оценка статьи на новостном портале, просмотр видео стриминговых сервисов — всё это является примером повседневного использования распределенных систем, которые де-факто являются неотъемлемой частью современных общественных отношений.

Распределенные системы представляют собой набор автономных вычислительных элементов, представляющихся пользователям (их клиентским устройствам) единой системой [1]. Взаимодействие клиента с распределенной системой выполняется путем обмена информацией. При этом, функционирование распределенной системы организовывается таким образом, чтобы возникающие сбои в процессе обмена информацией между ней и клиентом не приводили к возникновению противоречивого состояния обрабатываемых в ней данных. Природа сбоев разнообразна. Они возникают на клиентских устройствах, в сетях передачи данных, в самих компонентах распределенной системы (в следствии аппаратных сбоев, ошибок в программном обеспечении, ошибок в их настройке).

Операции взаимодействия с распределенной системой можно разделить на два класса: без изменения обрабатываемых системой данных (идемпотентные операции), с их изменением (не идемпотентные операции).

С учетом всего многообразия источников и причин сбоев, актуальной является задача проектирования API распределенной системы таким образом, чтобы при изменении обрабатываемых ею данных не идемпотентными операциями исключить риск нарушения консистентности, либо значительно снизить этот риск.

В настоящей работе рассматриваются способы проектирования API для управления не идемпотентными операциями в распределенных системах, взаимодействующих с клиентом по HTTP-протоколу. Дается сравнительный анализ подходов к организации API с защитой от неоднократного исполнения не идемпотентных операций. Предлагаются дополнительные функции, улучшающие характеристики идемпотентных API в распределенной системе.

Риски использования не идемпотентных операций

Из ранее приведенного определения распределенной системы следует, что обмен информацией в системе должен выполнятся гарантированно. В алгоритмах гарантированной доставки клиент направляет повторные запросы, если не получает от системы подтверждение о получении запроса по предыдущей попытке. Так, например, функционирует транспортный протокол TCP стека TCP/IP [2]. Неполучение клиентом подтверждения может возникнуть, когда запрос длительное время обрабатывается на промежуточных узлах и на клиенте наступает

Таблица 1. Сравнение идемпотентных API

Имя ключа	Stipe.com	Яндекс.Касса	Amazon EC2
	Idempotency-Key	Idempotence-Key	ClientToken
Сторона, формирующая ключ	Клиент	Клиент	Клиент
Рекомендуемый алгоритм формирования ключа	V4 UUID	V4 UUID	Строка длиной до 64 символов ASCII
Контроль запросов с разными параметрами для одного ключа	Есть	Не документировано	Есть
Длительность хранения ключа на стороне системы	24 часа	24 часа	Не документировано
HTTP-методы с использованием ключа	POST	POST	Не документировано
Дополнительные признаки области действия ключа идемпотентности	Отсутствуют	Отсутствуют	Область действия: зона в регионе, регион

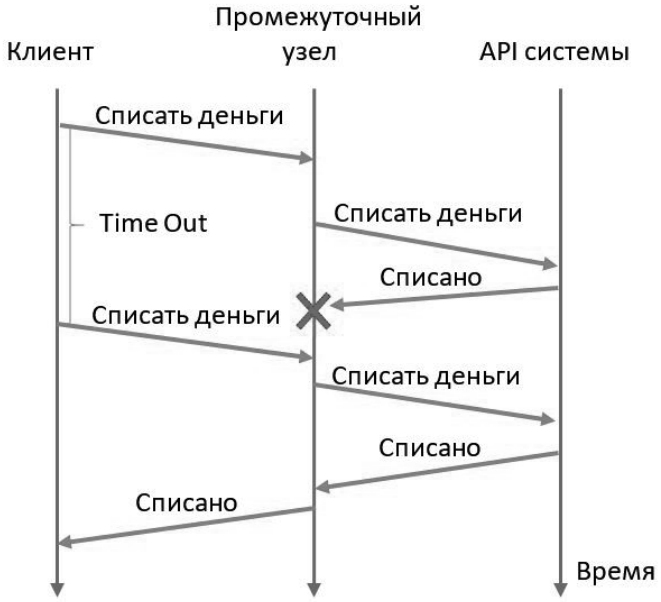

Рис. 1. Временная диаграмма повторного запроса не идемпотентной операции

превышение времени ожидания ответа. Также, возможна ситуация, когда запрос доставлен до системы, а ответ потерялся, либо вернулся клиенту с превышением времени ожидания ответа (рис. 1). При повторных запросах в перечисленных случаях идентичные не идемпотентные операции в системе выполняются множество раз [3].

Неоднократное исполнение идентичной не идемпотентной операции приводит к нарушению консистентности данных (в ситуации на рис. 1 денежные средства списываются дважды). При проектировании API распределенной системы требуется применение техник управления указанным риском.

Методы HTTP-протокола и их использование в API распределенных систем

Стандарт протокола HTTP/1.1 RFC7231 декларирует безопасные (GET, HEAD, OPTIONS, TRACE) и идем-

потентные методы (PUT, DELETE). По требованию стандарта их повторные вызовы не должны изменять состояния [4]. Стандарт RFC5789 дополняет протокол HTTP/1.1 методом PATCH, который, при определенных условиях использования, также декларируется идемпотентным [5]. Вместе с этим, не идемпотентным является метод POST.

При проектировании API распределенных систем возможна классификация операций над данными с позиции их влияния на состояние: безопасные, идемпотентные и не идемпотентные. В этом случае реализация каждой функции API возможна соответствующими методами HTTP-протокола.

Идемпотентные API, исключающие повторное исполнение не идемпотентных операций

Сервис управления платежами Stipe.com предлагает механизм, по которому не идемпотентные запросы выполняются как идемпотентные [6]. Технология защиты от повторного исполнения идентичных запросов заключается во включении клиентом в запрос уникального ключа Idempotency-Key. Сервис возлагает формирование значения уникального ключа на клиента. Поставщик сервиса рекомендует в качестве уникального ключа использовать универсальный уникальный идентификатор 4-й версии UUID [5]. Ключ Idempotency-Key допускается только в методе POST. При появлении ключа в методах GET и DELETE он игнорируется, т.к. оба метода по определению RFC7231 являются безопасным и идемпотентным. Сервис аннулирует известные ему ключи идемпотентности по истечении 24 часов с момента их первой регистрации.

В платежном сервисе Яндекс.Касса реализован аналогичных поход. Смысл идемпотентности API сервиса заключается в том, что многократные запросы обрабатываются так же, как однократные. Каждый POST-запрос дополняется заголовком Idempotence-Key. Особенностью реализации API заключаются в том, что если данные в запросе те же, а ключ идемпотентности отличается, то запрос выполняется как новый [7].

В сервисе Amazon Elastic Compute Cloud изменение состояния ресурса называется его мутацией. Методы API, приводящие к мутации ресурса, поддерживают защиту от случайного их многократного вызова. Методика защиты от повторного запуска аналогична рассмотренным выше. Отличием является специфика самого сервиса Amazon Elastic Compute Cloud: ключ идемпотентности расширен параметром области его действия — зона в регионе, регион со всеми включенными в него зонами [8].

Результаты сравнения (Таблица 1) показывают, что ответственность за исключение коллизий значений ключей идемпотентности возлагается на клиентов. Сервис управления платежами Stipe.com является наиболее документированным по сравнению с другими рассматриваемыми сервисами. Также, сервис Amazon EC2, в силу своей специфики, имеет дополнительные критерии применения уникальности ключа — признак локализации.

Предложения для дальнейшего развития идемпотентных API

Рассмотренные сервисы предоставляют идемпотентные API для запросов, использующих HTTP-метод POST. Известно, что запросы в режиме для чтения ресурса используют HTTP-метод GET. Например, таковым может быть запрос на получение произвольного объекта. В документации к сервисам Stipe.com, Яндекс.Касса и Amazon EC2 указано, что ключи идемпотентности применяются к HTTP-методу POST, но не GET. Из этого следует, что при вынужденном повторном запросе произвольного объекта защита не сработает. Такое поведение может привести к необоснованному росту счетчиков обращений к ресурсам. Так, например, необоснованно может вырасти индекс количества просмотров видеоролика на стриминговом сервисе, индекс количества оценок пользовательского сообщения в социальной сети.

Для исключения подобного нежелательного поведения предлагается использовать защиту с ключом идемпотентности для запросов ресурсов со строгим учетом количества обращений к ним.

Заключение

В статье показаны случаи нарушения согласованности данных, обрабатываемых в распределенных системах. Негативным примером противоречивых данных показано необоснованное двойное списание денежных средств.

Способом устранения рисков двойного исполнения одинакового запроса к распределенной системе является реализация его API с поддержкой идемпотентности.

В настоящей работе показана ситуация, в которой сервисы со строгим учетом количества обращений к ресурсам, вне зависимости от используемого HTTP-метода, должны использовать защиту ключом идемпотентности. Предлагаемый подход обеспечивает снижение риска возникновения противоречивого состояния данных в системе.

ЛИТЕРАТУРА

1. van Steen M., Tanenbaum A. S. Distributed Systems, 3rd ed., distributed-systems.net, 2017.— 582 P.
2. Хант К. TCP/IP. Сетевое администрирование. 3-е издание.— Пер. с англ.— СПб: Символ-Плюс, 2007.— 816 с.
3. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка.— СПб.: Питер, 2018.— 640 с.
4. Домашняя страница спецификации RFC4122 URL: https://tools.ietf.org/html/rfc4122 (дата обращения: 05.02.2020)
5. Домашняя страница спецификации RFC5789 URL: https://tools.ietf.org/html/rfc5789 (дата обращения: 05.02.2020)
6. Домашняя страница документации идемпотентных запросов платежного сервиса Stripe.com URL: https://stripe.com/docs/api/idempotent_requests (дата обращения: 05.02.2020)
7. Домашняя страница документации идемпотентных запросов платежного сервиса Яндекс.Касса URL: https://kassa.yandex.ru/developers/using-api/basics#idempotence (дата обращения: 05.02.2020)
8. Домашняя страница документации идемпотентных запросов Amazon Elastic Compute Cloud URL: https://docs.aws.amazon.com/AWSEC2/latest/APIReference/Run_Instance_Idempotency.html (дата обращения: 05.02.2020)

Журнал «Современная наука: актуальные проблемы теории и практики»

Г. Новосибирск