Санкт-Петербургский государственный университет

Факультет прикладной математики – процессов управления

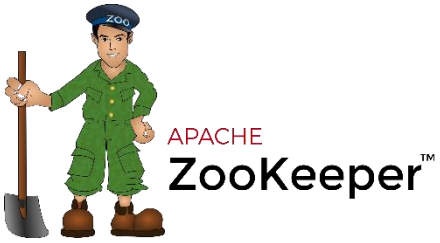
2022 год.

Apache ZooKeeper

Работу выполнил Панюшин Даниил Васильевич группа 19.Б12-пу.

Содержание

Определение



ZooKeeper — это централизованная служба для хранения информации о конфигурации, именования, обеспечения распределенной синхронизации и предоставления групповых операций. Все эти виды услуг в той или иной форме используются распределенными приложениями. Каждый раз, когда они создаются, требуется исправлять ошибки и условия гонки, появление которых неизбежно. Из-за сложности реализации такого рода служб приложения изначально обычно экономят на них, что делает их неустойчивыми при наличии изменений и сложными в управлении. Даже если все сделано правильно, разные реализации этих служб приводят к сложностям с управлением при развертывании.

ZooKeeper стремится объединить сущность этих различных сервисов в очень простой интерфейс для централизованной координации. Сам сервис является распределенным и очень надежным. Используемые протоколы реализованы самой службой, поэтому приложениям не нужно будет реализовывать их самостоятельно.

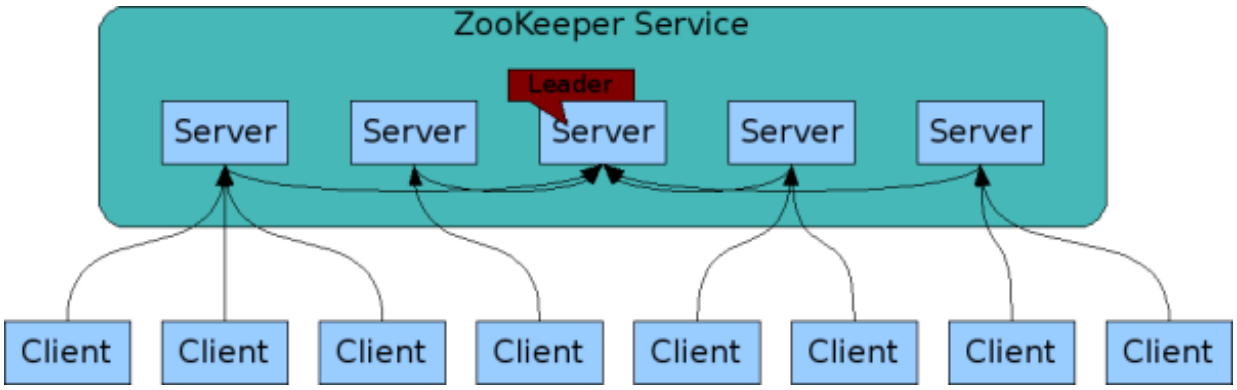
Структура

ZooKeeper позволяет распределенным процессам координировать свои действия друг с другом через общее иерархическое пространство имен регистров данных (мы называем эти регистры узлами), что очень похоже на файловую систему. В отличие от обычных файловых систем ZooKeeper обеспечивает своим клиентам высокую пропускную способность, низкую задержку, высокую доступность и строго упорядоченный доступ к узлам. Производительность ZooKeeper позволяет использовать его в больших распределенных системах. Надежность не позволяют ему стать единственной точкой отказа в больших системах. Упорядоченность позволяет реализовать на клиенте сложные примитивы синхронизации.

Имя в пространстве имен, предоставленным ZooKeeper, представляет собой последовательность элементов пути, разделенных косой чертой ("/"). Каждый узел в пространстве имен идентифицируется путем. У каждого узла есть родитель, путь которого является префиксом узла с одним элементом меньше. Исключением из этого правила является корневой узел ("/"), у которого нет родителя. Кроме того, точно так же, как и в стандартныех файловых системах, узел нельзя удалить, если у него есть дочерние элементы.

Основное различие между ZooKeeper и стандартной файловой системой заключается в том, что каждый узел может иметь связанные с ним данные (каждый файл также может быть каталогом и наоборот). При этом узлы ограничены по объему данных, которые они могут иметь.

ZooKeeper был разработан для хранения координационных данных: информации о состоянии, конфигурации, местоположении и т. д. Существует встроенная проверка того, не превышает ли потребляемый объём памяти, выделенной под метаинформацию, один мегабайт. Это ограничение необходимо чтобы предотвратить использование ZooKeeper в качестве большого хранилища данных.



ZooKeeper реплицируется на набор компьютеров, составляющих службу. Эти машины поддерживают образ дерева данных в памяти вместе с журналами транзакций и моментальными снимками (с английского snapshot) в постоянном хранилище. Поскольку информация хранятся в памяти, ZooKeeper обеспечивает очень высокую пропускную способность и низкую задержку. Недостатком хранения базы данных в памяти является то, что её размер ограничен объемом памяти, что является еще одной причиной для хранения небольшого объема данных на узлах.

Серверы, составляющие службу ZooKeeper, должны знать друг о друге. Пока доступно большинство серверов, служба ZooKeeper будет доступна. Клиенты также должны знать список серверов.

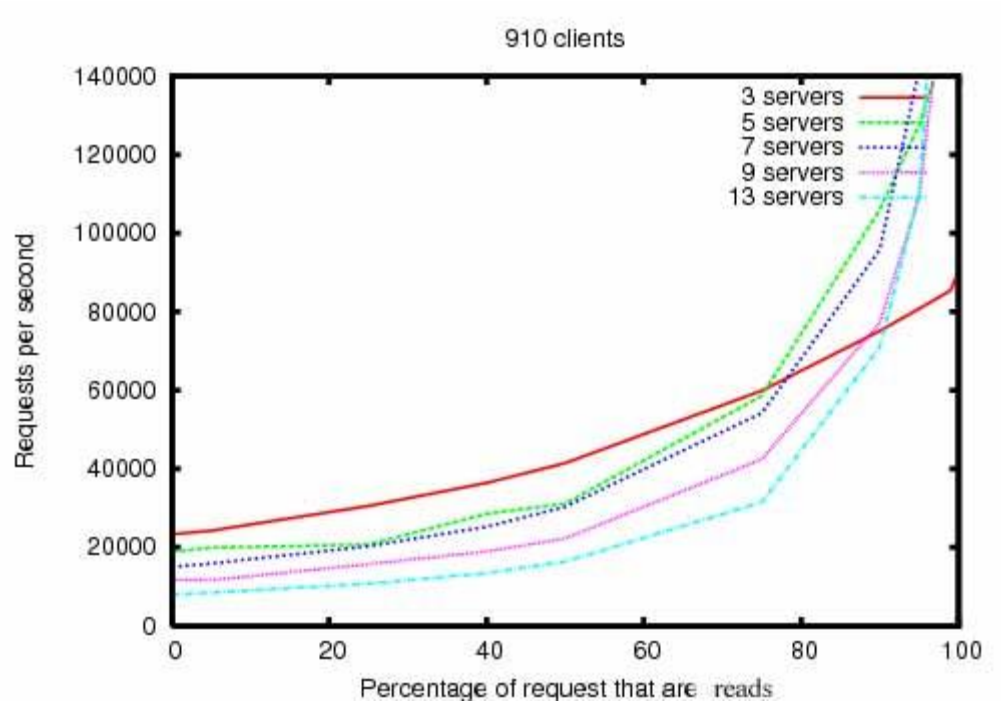
Клиенты подключаются только к одному серверу ZooKeeper. Клиент поддерживает TCP-соединение, через которое он отправляет запросы, получает ответы, получает события триггеры и отправляет тактовые импульсы. Если TCP-соединение с сервером разорвется, клиент подключится к другому серверу. Когда клиент впервые подключается к службе ZooKeeper, первый доступный сервер устанавливает сеанс для клиента. Если клиенту необходимо подключиться к другому серверу, то сеанс будет заново создан с новым сервером.

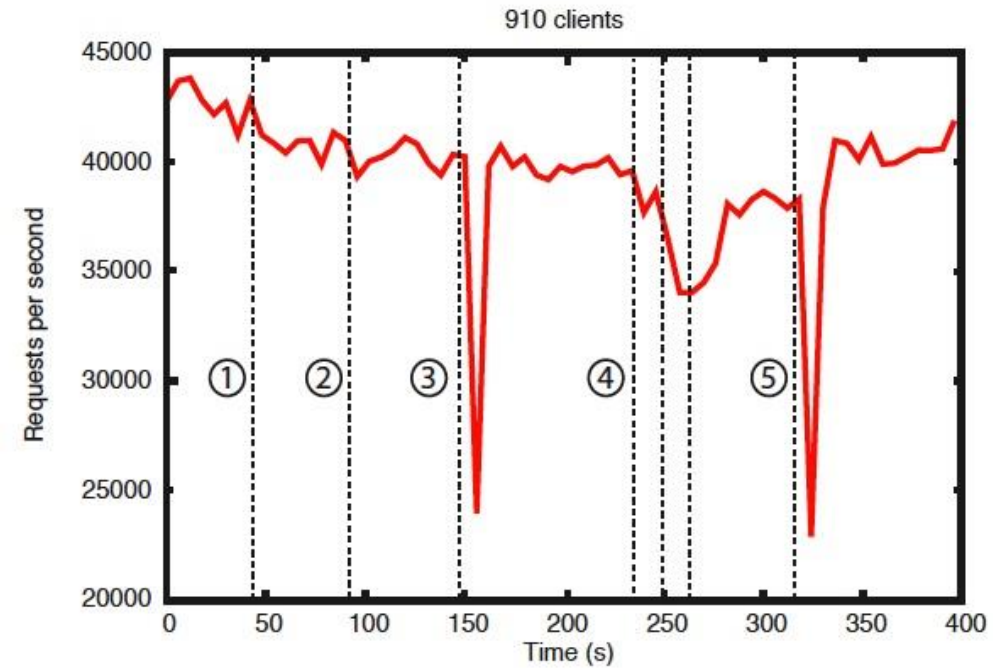
Запросы на чтение, отправленные клиентом ZooKeeper, обрабатываются локально на сервере, к которому подключен клиент. Если запрос на чтение регистрирует триггер, отслеживающий узел, то этот триггер также отслеживается локально на сервере ZooKeeper. Запросы на запись перенаправляются на другие серверы и проходят согласование, прежде чем будет сгенерирован ответ. Запросы на синхронизацию также перенаправляются на другой сервер. Таким образом, пропускная способность запросов на чтение масштабируется с количеством серверов, а пропускная способность запросов на запись уменьшается с количеством серверов.

Все обновления полностью упорядочены. ZooKeeper фактически помечает каждое обновление идентификатором, который отражает этот порядок. Каждое обновление будет иметь уникальный идентификатор. Операции чтения упорядочены относительно обновлений. Ответы на запросы на чтение будут помечены последним идентификатором, обработанным сервером, обслуживающим чтение.

Зачем использовать ZooKeeper?

Как было сказано выше, ZooKeeper обладает высокой производительностью и устойчивостью при высоких нагрузках.





ZooKeeper, поскольку его целью является создание основы для построения более сложных сервисов, предоставляет набор гарантий:

- Последовательная согласованность — обновления от клиента будут применяться в том порядке, в котором они были отправлены.
- Атомарность — обновления либо завершаются успешно, либо завершаются ошибкой. Промежуточных результатов нет.
- Единый образ системы — клиент увидит одно и то же представление службы независимо от сервера, к которому он подключается. т. е. клиент никогда не увидит более старый образ системы, даже если клиент переключится на другой сервер под тем же сеансом.
- Надежность. После применения обновления оно будет храниться до тех пор, пока клиент не перезапишет обновление.

- Своевременность. Представление клиентов о системе гарантированно будет актуальным в течение определенного периода времени.

Отдельно стоит отметить простоту интерфейса взаимодействия с ZooKeeper.

Недостатки

Слабостью ZooKeeper является тот факт, что произошедшие изменения сбрасываются: поскольку триггеры одноразовые и существует задержка между получением события и отправкой нового запроса на создание триггера, мы не можем гарантированно увидеть каждое изменение, которое происходит с узлом ZooKeeper. Необходимо обрабатывать такие случаи - когда узел меняется несколько раз между получением события и повторной установкой триггера. Таким образом, ZooKeeper основывается не на событиях, а на состояниях, то есть нельзя использовать события для регистрации когда и как что-то поменялось – для этого история изменения должна быть включена в данные.

Пример системы, использующей ZooKeeper

Давайте рассмотрим один пример того, где может быть полезен ZooKeeper. Представьте себе некоторую медиа платформу - сложную серверную систему, работающую, скажем, на 100 узлах в кластере. Подсистемы, которые должны работать на этих узлах: база данных, мониторинг, детекторы мошенничества, серверы-маяки, обработчики журнала событий веб-сервера, информационные панели клиентов, планировщики кампаний, тестеры сценариев кампаний, обновления, установки, медиа-менеджеры и так далее.

Теперь представьте, что все машины включаются. Как все процессы на всех хостах узнают, что делать? А если несколько машин выходят из строя, то как все процессы узнают, что делать в этой ситуации? Здесь в дело вступает служба координации. На ней все эти подсистемы определяют, что они должны делать по отношению ко всем другим подсистемам в продукте.

Например, как серверы, раздающие контент, узнают, какую базу данных использовать? Понятно, что вариантов решения этой проблемы много. Одним из них является использование стандартных DNS соглашений об именах. А вот в случае с конфигурационными файлами жесткое кодирование по-прежнему является фаворитом. Использование локатора служб начальной загрузки — еще один вариант (при условии, что есть возможность выполнить начальную загрузку локатора).

В идеале любое решение должно одинаково хорошо работать во время модульного тестирования, системного тестирования и развертывания. В этом случае ZooKeeper действует как локатор сервисов. Каждый процесс обращается к ZooKeeper и выясняет, какая база данных является первичной. Если выбран новый первичный сервер, например, из-за сбоя хоста, ZooKeeper отправляет событие, которое позволяет всем, кто зависит от базы данных, отреагировать, получив новую первичную базу данных. Сложная логика повторных попыток в коде приложения для переключения на другой сервер базы данных — это излишне сложное решение. Использование ZooKeeper прекрасно решает проблему обнаружения сервисами других сервисов во всех сценариях.

Как серверы баз данных решают, какую роль они будут играть в первую очередь? Все серверы баз данных загружаясь должны понимать, какую роль они играют в системе – первичный или вторичный сервер. База данных может быть шардом, а значит она должна понимать какой

диапазон ключей обслуживать. Если 10 серверов являются серверами баз данных, согласование ролей может быть очень сложным и подверженным ошибкам процессом. Для решения этой проблемы можно позволить серверам баз данных включаться и самоорганизовываться при первоначальном выборе роли и в сценариях сбоя. Преимущество такой системы в том, что она может работать локально на одной машине или на множестве машин внутри кластера с минимальными усилиями. ZooKeeper поддерживает такой тип координирования.

Теперь предположим, что нужно изменить конфигурацию некоторых сервисов, работающих в настоящее время в 40 процессах на 40 разных хостах. Как это сделать? Первый подход – это отсутствие подхода. Выпуск новой версии кода решает эту проблему, но очень медленно. Другой подход — файл конфигурации. Конфигурация помещается в дистрибутив и рассылается на все узлы. Затем каждый сервис периодически проверяет, не изменился ли файл конфигурации, и если да, то новая конфигурация считывается. Сложность заключается в том, что необходимо знать, какие пакеты выполняются на каких узлах. Так же нужно настроить систему откатов назад, если не все пакеты отправляются правильно. Это сложно, а когда вносятся изменения, влияющие на несколько подсистем, то все становится еще сложнее. Данные проблемы решаются применением ZooKeeper в качестве системы координирования.

Кто использует ZooKeeper?

ZooKeeper является проектом Apache Software Foundation. На нем основываются многие другие продукты, например, Apache Kafka.

Данная система используется в таких компаниях, как:

- Yahoo!
- VK
- Meta
- eBay
- Reddit

Использованная литература.

- https://cwiki.apache.org/confluence/display/ZOOKEEPER/ProjectDescription
- http://highscalability.com/blog/2008/7/15/zookeeper-a-reliable-scalable-distributed-coordination-syste.html
- https://zookeeper.apache.org/doc/current/zookeeperOver.html#sc_dataModelNameSpace
- https://javarush.ru/groups/posts/2162-zookeeper-ili-kak-zhivjetsja-rabotniku-zooparka
- https://en.wikipedia.org/wiki/Apache_ZooKeeper