Programming Assignment: Клиент для отправки метрик

✔ Passed · 1/1 points

Deadline The assignment was due on April 1, 12:59 AM PDT
You can still pass this assignment before the course ends.

Instructions My submission Discussions

Хранение метрик

Если вы разрабатываете настоящий проект, у которого есть большое количество пользователей, то необходимо наблюдать за всеми процессами, происходящими в нем. Для этого нужно смотреть за численными показателями в проекте. Показатели могут быть самыми разными - количество запросов к вашему приложению, время ответа вашего сервиса на каждый запрос, количество пользователей в сутки, и т.д. Эти всевозможные численные показатели мы будем называть метриками.

Для сбора, хранения и отображения подобных метрик существуют готовые решения, например Graphite, InfluxDB. Мы в рамках курса разработаем свою систему для сбора метрик - сервер и клиент.

На этой неделе мы начнем с разработки клиента для отправки подобных метрик на сервер, где они хранятся, и могут быть запрошены в любой момент времени. Затем в качестве финального задания на шестой неделе вам будет предложено реализовать и сам сервер.

Протокол взаимодействия

Итак, на этой неделе вам необходимо разработать сетевую программу-клиент, при помощи которой можно отправлять различные метрики на сервер. Клиент и сервер должны взаимодействовать между собой по простому текстовому протоколу через TCP сокеты. Текстовый протокол имеет главное преимущество – он наглядный – можно просмотреть диалог взаимодействия клиентской и серверной стороны без использования дополнительных инструментов.

Прежде чем реализовывать клиентское приложение давайте рассмотрим взаимодействие между клиентом и сервером на конкретных примерах.

Предположим, необходимо собирать метрики о работе операционной системы: cpu (загрузка процессора), memory usage (потребление памяти), disk usage (потребление места на жестком диске), network usage (статистика сетевых интерфейсов) и т.д. Это понадобится для контроля загрузки серверов и прогноза по расширению парка железа компании - проще говоря для мониторинга.

Пусть у нас имеется в наличии два сервера palm и eardrum. Мы будем получать загрузку центрального процессора на сервере и отправлять метрику с названием имя_сервера.cpu

```
client -> server: put palm.cpu 10.6 1501864247\n

server -> client: ok\n\n

client -> server: put eardrum.cpu 15.3 1501864259\n

server -> client: ok\n\n
```

Чтобы отправить метрику на сервер, вы отправляете в TCP-соединение строку вида:

put palm.cpu 10.6 1501864247\n

Ключевое слово put означает команду отправки метрики. За ней через пробел следует название (имя) самой метрики, например palm.cpu, далее опять через пробел значение метрики, и через еще один пробел временная метка unix timestamp. Таким образом, во время 1501864247 значение метрики palm.cpu было равно 10.6. Наконец, команда заканчивается символом переноса строки \n.

В ответ на эту команду put сервер присылает уведомление об успешном сохранении метрики в виде строки:

ok\n\n

Два переноса строки в данном случае означают маркер конца сообщения от сервера клиенту.

Команды

Необходимо реализовать две команды:

put - для сохранения метрик на сервере.

get - для получения метрик.

Формат команды put для отправки метрик — это строка вида:

put <key> <value> <timestamp>\n

Успешный ответ от сервера:

ok\n\n

Ошибка сервера:

error\nwrong command\n\n

Обратите внимание на то, что за каждым ответом сервера указано два символа \n. В качестве значения метрики value используется вещественное число.

Данные нужно не только отправлять на сервер, но и запрашивать их. Это может потребоваться для визуализации и анализа нужных метрик в определенные промежутки времени.

Формат команды get для получения метрик — это строка вида:

get <key>\n

В качестве ключа можно указывать символ *, для этого символа будут возвращены все доступные метрики. В данном задании мы никак не ограничиваем количество метрик, которые должен вернуть сервер – сервер должен возвращать все метрики, удовлетворяющие ключу.

Успешный ответ от сервера:

ok\npalm.cpu 10.5 1501864247\neardrum.cpu 15.3 1501864259\n\n

Если ни одна метрика не удовлетворяет условиям поиска, то вернется ответ:

ok\n\n

Обратите внимание, что каждая успешная операция начинается с "ok", а за ответом сервера всегда указано два символа \n.

Реализация клиента.

Необходимо реализовать класс Client, в котором будет инкапсулировано соединение с сервером, клиентский сокет и методы для получения и отправки метрик на сервер. В конструктор класса Client должна передаваться адресная пара хост и порт, а также необязательный аргумент timeout (timeout=None по умолчанию). У класса Client должно быть 2 метода: put и get, соответствующих протоколу выше.

Пример вызова клиента для отправки метрик и затем их получения:

```
client = Client("127.0.0.1", 8888, timeout=15)

client.put("palm.cpu", 0.5, timestamp=1150864247)
client.put("palm.cpu", 2.0, timestamp=1150864248)
client.put("palm.cpu", 0.5, timestamp=1150864248)

client.put("eardrum.cpu", 3, timestamp=1150864250)
client.put("eardrum.cpu", 4, timestamp=1150864251)
client.put("eardrum.memory", 4200000)

print(client.get("*"))
```

Клиент получает данные в текстовом виде, метод get должен возвращать словарь с полученными ключами с сервера. Значением ключа в словаре является список кортежей [(timestamp, metric_value), ...], отсортированный по timestamp от меньшего к большему. Значение timestamp должно быть преобразовано к целому числу int. Значение метрики metric_value нужно преобразовать к числу с плавающей точкой float.

Метод put принимает первым аргументом название метрики, вторым численное значение, третьим - необязательный именованный аргумент timestamp. Если пользователь вызвал метод put без аргумента timestamp, то клиент автоматически должен подставить текущее время в команду put - str(int(time.time()))

Метод put не возвращает ничего в случае успешной отправки и выбрасывает исключение ClientError в случае неуспешной.

Метод get принимает первым аргументом имя метрики, значения которой мы хотим выгрузить. Также вместо имени метрики можно использовать символ *, о котором говорилось в описании протокола.

Метод get возвращает словарь с метриками (смотрите ниже пример) в случае успешного получения ответа от сервера и выбрасывает исключение ClientError в случае неуспешного.

Пример возвращаемого значения при успешном вызове client.get("palm.cpu"):

```
{
  'palm.cpu': [
    (1150864247, 0.5),
    (1150864248, 0.5)
  ]
}
```

Пример возвращаемого значения при успешном вызове client.get("*"):

```
{
  'palm.cpu': [
    (1150864247, 0.5),
    (1150864248, 0.5)
  ],
  'eardrum.cpu': [
    (1150864250, 3.0),
    (1150864251, 4.0)
  ],
  'eardrum.memory': [
    (1503320872, 4200000.0)
  ]
}
```

Если в ответ на get-запрос сервер вернул положительный ответ ok\n\n, но без данных (то есть данных по запрашиваемому ключу нет), то метод get клиента должен вернуть пустой словарь:

```
>>> client.get("non_existing_key")
{}
```

Обратите внимание, что сервер хранит данные с максимальным разрешением в одну секунду. Это означает, что если в одну и ту же секунду отправить две одинаковые метрики, то будет сохранено только одно значение, которое было обработано последним. Все остальные значения будут перезаписаны.

Итак, вам необходимо предоставить модуль с классом Client, исключением ClientError. В этом классе Client должны быть доступны методы get и put с описанной выше сигнатурой. При вызове методов get и put клиент должен посылать сообщения в TCP-соединение с сервером в соответствии с описанным текстовым протоколом, получать ответ от сервера, преобразовывать его в удобный для использования формат, описанный выше.

Код клиента неудобно разрабатывать и отлаживать без сервера. Для удобства тестирования во время разработки кода клиента мы разработали unittest-ты.

test_week5.py

Используйте данный unittest для проверки работы Вашего клиента для отправки метрик. Это ускорит процесс разработки клиента и упростит отладку. Проверяйте работу перед отправкой решения на оценку.

Успехов при выполнении задания!

How to submit

When you're ready to submit, you can upload files for each part of the assignment on the "My submission" tab.