Asynckafka Documentation

Release

José Melero Fernández

Mar 07, 2018

Contents:

CHAPTER 1

Asynckafka

Fast python kafka client for asyncio. Asynckafka is written in Cython on top of Rdkafka as kafka driver.

Right now it is work in progress, so use it at our own risk. Before the 1.0.0 release i don't warranty stability in the api between the minor version numbers.

Documentation url: WIP

1.1 Performance

This project was born from the need to have a high performance kafka library for asyncio.

1.1.1 Benchmark

Simple benchmark with one kafka broker, one partition, 200 bytes per message and 10 millions of messages:

```
Preparing benchmark.
Filling topic  benchmark_ad5682b7-9469-4f35-ad72-933c5e9879e1 with
10000000 messages of 200 bytes each one.
The time used to produce the messages is 21.905211210250854 seconds.
Throughput: 91.30247505050632 mb/s

Starting to consume the messages.
The time used to consume the messages is 20.685954093933105 seconds.
Throughput: 96.68396202167787 mb/s
```

CHAPTER 2

Requirements

1. Python 3.6 or greater
2. Rdkafka 0.11.X

2.1 Install rdkafka from source

You need the Rdkafka headers to be able to compile asynckafka, download rdkafka from here, then unpack the package and run:

```
./configure
make
sudo make install
```

2.2 Install asynckafka package

The package is in pypi, you can install it with pip:

```
$ pip install asynckafka
```

CHAPTER 3

Examples

3.1 Simple consumer

```
import asyncio
import logging
import sys

from asynckafka import Consumer

logging.basicConfig(stream=sys.stdout, level=logging.DEBUG)


async def consume_messages(consumer):
    async for message in consumer:
        print(f"Received message: {message.payload}")

consumer = Consumer(
    brokers='localhost:9092',
    topics=['my_topic'],
    group_id='my_group_id',
)
consumer.start()

asyncio.ensure_future(consume_messages(consumer))

loop = asyncio.get_event_loop()
try:
    loop.run_forever()
finally:
    consumer.stop()
    loop.stop()
```

3.2 Simple producer

```
import asyncio
import logging
import sys

from asynckafka import Producer

logging.basicConfig(stream=sys.stdout, level=logging.DEBUG)


async def send_messages(producer):
    while True:
        await producer.produce("my_topic", b"my_message")
        print('sent message')
        await asyncio.sleep(1)

producer = Producer(brokers="localhost:9092")
producer.start()

asyncio.ensure_future(send_messages(producer))

loop = asyncio.get_event_loop()

try:
    loop.run_forever()
finally:
    producer.stop()
    loop.stop()
```

CHAPTER 4

RdKafka configuration

This library is built on top of rdkafka, and as wrapper it uses the same configuration that it. The rdkafka configuration is very extensive and can be found in the following link:

https://github.com/edenhill/librdkafka/blob/v0.11.3/CONFIGURATION.md

The asynckafka producer and consumer accepts two configuration dicts. The dicts be strings in the key and value.

Consumer configuration, example:

```
Consumer(
    brokers='127.0.0.1:9092',
    topics=['my_topic'],
    rdk_consumer_config={
        'api.version.request': 'true'
        'enable.auto.commit': 'false'
    },
    rdk_topic_config={
        'auto.offset.reset':  'smallest'
    }
)
```

Producer configuration, example:

```
producer = Producer(
    brokers='127.0.0.1:9092',
    rdk_producer_config={
        'batch.num.messages': '100000',
        'message.send.max.retries': 4,
    },
    rdk_topic_config={
        'message.timeout.ms': '10000'
    },
)
```

The configuration *rdk_consumer_config* correspond with those that can be found at 'Global configuration properties', section of rdkafka configuration documentation (less the exclusive configuration from the producer, indicated with a

'P' to the left of the name). The rdk_producer_config is the opposite of the consumer, so

you should exclude the ones indicated with 'C'.

At the end of configuration you can find the 'Topic configuration properties', there is located the configuration that can be passed to the argument *rdk_topic_config* of the consumer of producer.

CHAPTER 5

Logging

Asynckafka uses the standard python logging library, with “asynckafka” as logger.

To enable all the logging to stdout it is enough with:

```
import logging
import sys

logging.basicConfig(stream=sys.stdout, level=logging.DEBUG)
```

There are some python logger debug lines disabled by default in the consumer and producer. The reason is avoid python calls and python strings composition for the python logger in the critical path of cython. Anyway, you can enable them with:

```
import asynckafka

asynckafka.set_debug(True)
```

CHAPTER 6

Error callback

The error callback can be passed to the consumer of producer, it should be a coroutine function and accepts one parameter. This parameter is a KafkaError. This error_callback is thread safe and it is executed in the loop used by the consumer or producer.

Example:

```
async def error_callback(kafka_error):
    print(kafka_error)

# Should be a wrong port
self.producer = Producer(
    brokers="127.0.0.1:6000",
    error_callback=error_callback
)
```

CHAPTER 7

Develop

7.1 How to run the tests

WIP

CHAPTER 8

API

8.1 Consumer

8.2 Producer

8.3 Message

class asynckafka.**Messsage**

payload

8.4 Kafka Error

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Index