# RbFly Library Performance Report

***Release 0.8.3***

**Artur Wroblewski**

**May 04, 2024**

## Contents

**1 Publishing Messages** . . . **2**
1.1 Results . . . 2
1.2 Test Scenario . . . 2

**2 Publishing Messages in AMQP Format** . . . **3**
2.1 Results . . . 3
2.2 Test Scenario . . . 3

**3 Encoding and Decoding with AMQP Codec** . . . **4**
3.1 Results . . . 4
3.2 Test Scenario . . . 4

**4 Testing Environment** . . . **5**
4.1 Software Components . . . 5
4.2 Hardware Configuration . . . 5

RbFly library is implemented with performance in mind. For example

- RbFly can be over 3.5 times faster than other, similar solutions when publishing messages to a streaming broker (see Section 1.1)
- RbFly's AMQP 1.0 codec can be over 37 times faster compared to other AMQP 1.0 codecs (see Section 3.1)

Performance of RbFly is achieved by implementing critical sections of code using Cython, avoiding copying of data, and testing. The testing involves use of asyncio drop-in replacement uvloop, which improves performance of Python asyncio based applications.

Next sections of the report describe the results of the performance tests executed during RbFly project development

- throughput of publishing messages to a streaming broker
- throughput of publishing messages in AMQP 1.0 format to a RabbitMQ stream
- throughput of encoding and decoding messages using AMQP 1.0 codec

**Note:** Results of performance testing might be affected by bias of a test scenario, or local conditions. Please submit a ticket in RbFly project's issues system to propose an improvement to the performance testing procedures, the scripts, and the documentation of the project.

# 1 Publishing Messages

The test measures the performance of sending messages to a streaming broker. There are two subtests

- publish single message, and receive confirmation from a broker for the message
- publish batch of 100 messages, and receive confirmation from a broker for the batch

Section 1.2 outlines the testing procedure.

RbFly performance is tested against Python *asyncio* libraries for streaming technologies like RabbitMQ Streams, or Apache Kafka, see Section 4.1 for additional details.

## 1.1 Results

The combination of RbFly library and RabbitMQ Streams broker can publish messages two times faster (single message) or over 3.5 times faster (batch of 100 messages) comparing to similar solutions for Python language.

The summary of message publishing performance test is presented on Figure 1.1.

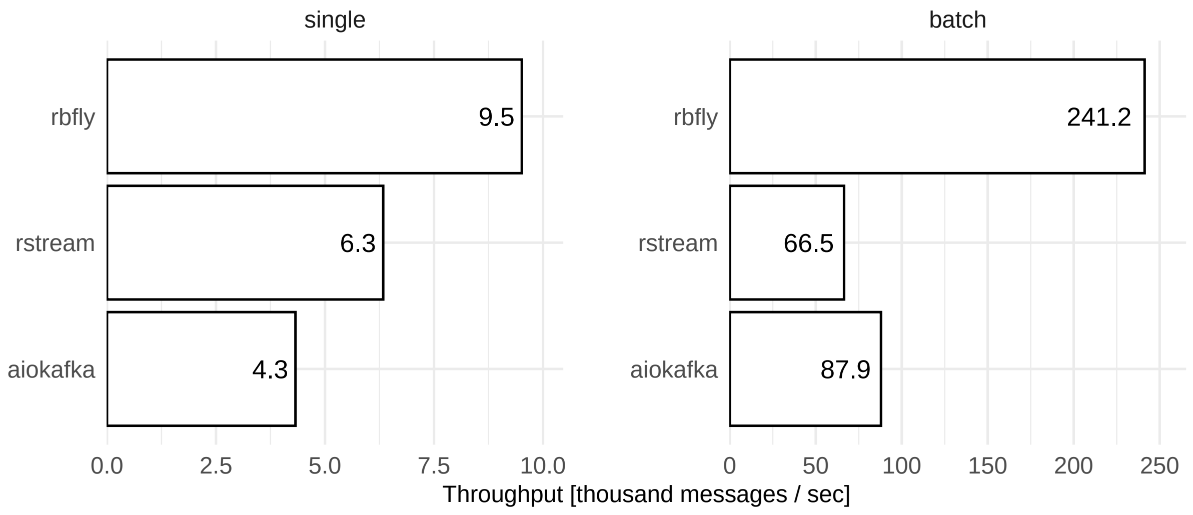

Figure 1.1: Performance of message publishing by RbFly, and related libraries
Median number of published messages per second for RbFly, *rstream*, and *aiokafka* libraries. The libraries send single message (left panel) or batch of 100 messages (right panel).

## 1.2 Test Scenario

The performance testing scenario is as follows

1. Publish 200 thousand messages to a broker. Receive a confirmation from the broker for each message - single message publishing test.
2. Publish 1 million messages, in batches of 100 messages, to a broker. Receive a confirmation from the broker for each batch.
3. Each message is 10 bytes.
4. RabbitMQ Streams client libraries publish messages in AMQP format.
5. Apache Kafka client libraries publish messages using MessagePack binary format.
6. Repeat each round of testing 100 times.
7. Calculate median value of number of messages published to a broker, and present as a bar plot.

# 2 Publishing Messages in AMQP Format

The test measures the performance of publishing messages using RbFly project's AMQP encoder. This is compared against sending simple binary string messages. There are two subtests

- publish single message, and receive confirmation from a broker for the message
- publish batch of 100 messages, and receive confirmation from a broker for the batch

## 2.1 Results

RbFly's AMQP encoder affects throughput of publishing messages to RabbitMQ Streams broker. However, no significant difference can be observed when sending messages in AMQP format compared to sending messages as binary string.

The summary of the performace test of publishing messages in AMQP format is presented on Figure 2.1.

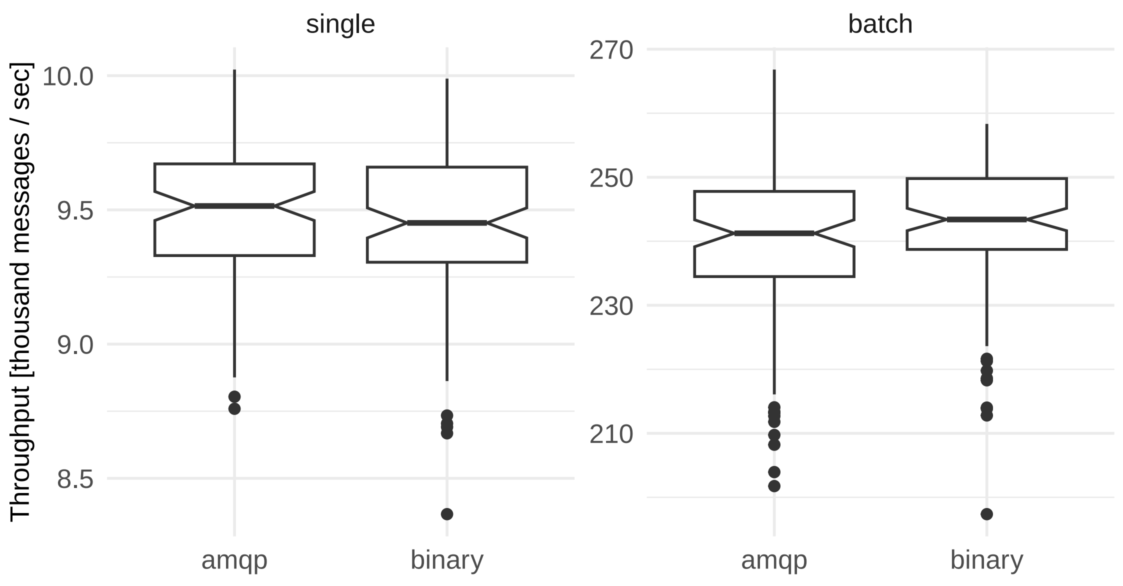

Figure 2.1: Throughput of publishing messages in AMQP format and as a binary string
Median number of published messages per second using RbFly's AMQP encoder and as a binary string. RbFly sends single message (left panel) or batch of 100 messages (right panel).

## 2.2 Test Scenario

The scenario for performance testing of publishing messages in AMQP format is as follows

1. Publish 200 thousand messages to a broker. Receive a confirmation from the broker for each message - single message publishing test.
2. Publish 1 million messages, in batches of 100 messages, to a broker. Receive a confirmation from the broker for each batch.
3. Repeat each scenario by publishing message in AMQP format, or using a binary string.
4. Each message is a string of length 10 before encoding as AMQP, or as a binary string.
5. Repeat each round of testing 100 times.
6. Calculate median value of number of messages published to a broker, and present as a box plot.

A binary string for a test is generated using consecutive numer of a message, for example in Python language:

```
'{:010d}'.format(1357).encode()
```

which encodes the binary string:

```
b'0000001357'
```

## 3 Encoding and Decoding with AMQP Codec

The test measures performance of encoding and decoding data with AMQP 1.0 codecs implemented by RbFly, uAMQP, and Qpid Proton libraries (see also Section 4.1).

Section 3.2 outlines the testing procedure.

### 3.1 Results

RbFly's AMQP 1.0 codec is order of magnitude faster than codecs implemented by uAMQP and Qpid Proton projects.

The summary of the performace test of AMQP 1.0 codecs is presented on Figure 3.1.

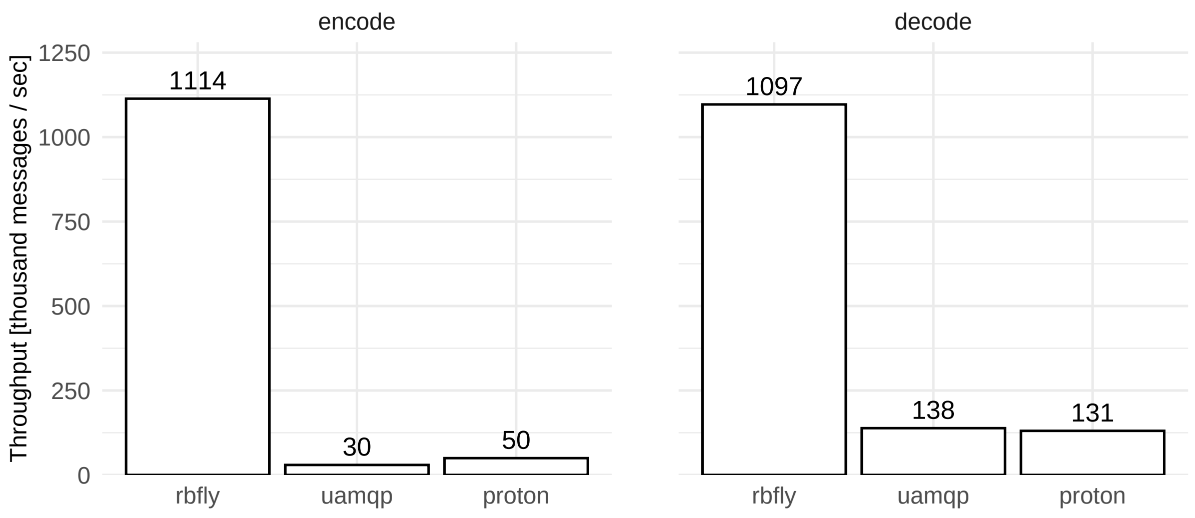

Figure 3.1: Performance of encoding and decoding of message data with AMQP codecs
Number of encoded (left panel) and decoded (right panel) messages per second by AMQP 1.0 codecs implemented by RbFly, uAMQP, and Qpid Proton projects.

### 3.2 Test Scenario

The performance test of AMQP 1.0 codecs processes data of a single AMQP 1.0 message. A message is encoded and decoded 10 thousand times. The execution time is measured with Python's `timeit.timeit()` function, and normalized to a number of processed messages per second.

The data of a AMQP 1.0 message is a dictionary, which consists of strings, binary strings, and integers:

```
{
    'large-int': 2 ** 32 + 1,
    'small-list': [1, 2, 3],
    'bin-string': b'abc',
}
```

# 4 Testing Environment

## 4.1 Software Components

The following software components are used during RbFly project performance testing

***aiokafka 0.8.1***
aiokafka is a client library for Apache Kafka distributed event streaming platform.

***Apache Kafka 3.4.0***
Apache Kafka is a distributed event streaming platform.

1. The broker is run with Zookeeper.
2. Its configuration option `log.dirs` is set to `/var/tmp/kafka-logs` - the directory points to permanent storage.

***Qpid Proton 0.39.0***
Qpid Proton is a messaging library, which integrates with AMQP 1.0 ecosystem from any platform.

***RabbitMQ 3.12.6***
RabbitMQ Streams is a streaming solution, which is part of RabbitMQ message broker project.

***rstream 0.13.0***
rstream is a Python client library for RabbitMQ Streams.

***uAMQP 1.6.5***
uAMQP is an AMQP 1.0 library for Python.

## 4.2 Hardware Configuration

The performance tests are executed on a laptop with the following configuration

- Intel CPU i5-1135G7, 2.40GHz
- 32 GB RAM
- Samsung SSD 970 EVO Plus 1TB

When running the performance tests

- the Intel's Turbo Boost Technology is switched off
- Linux CPU scaling governor is set to performance