

# Когда всё пошло по Kafka 2

Разгоняем продьюсеров

Григорий Кошелёв  
Контур

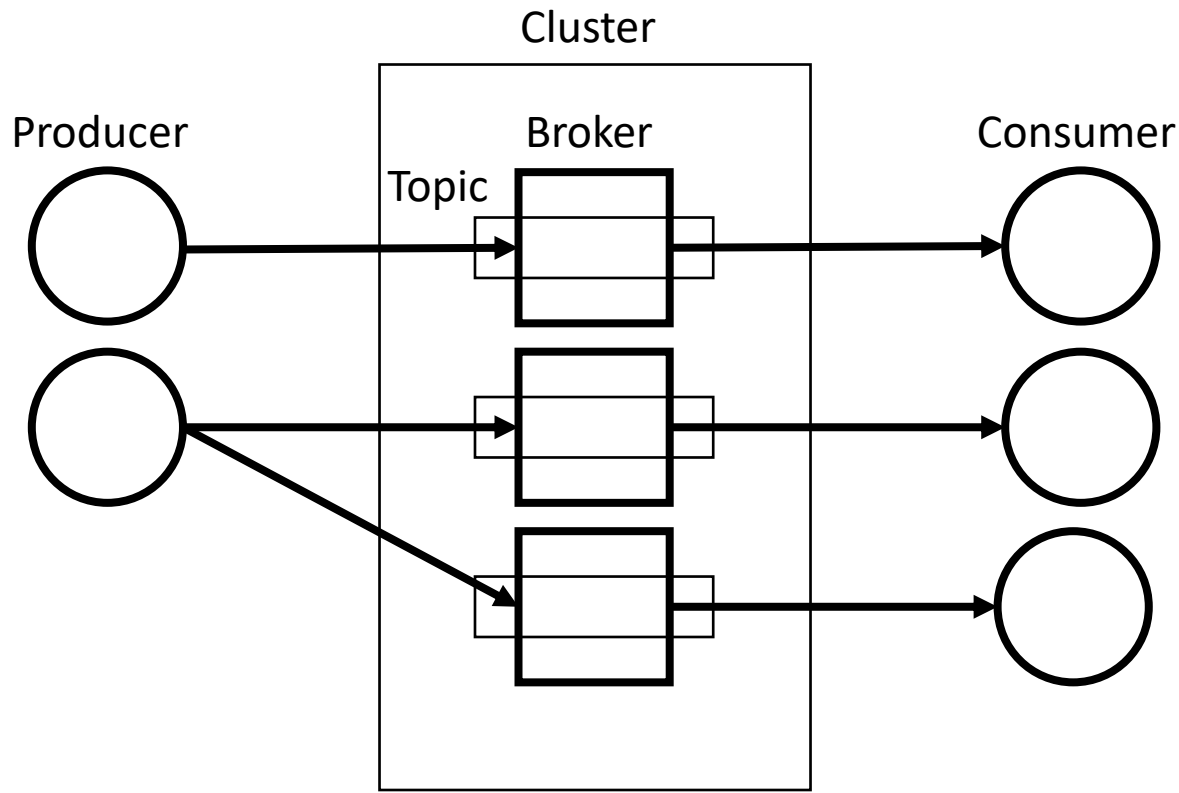
# Что будет в докладе

- Устройство Kafka Producer
- Настройки
  - Производительность
  - Внешние ограничения (например, гарантия сохранности)
- Метрики производительности
- Поиск узких мест
- Тюнинг производительности

# Мотивация

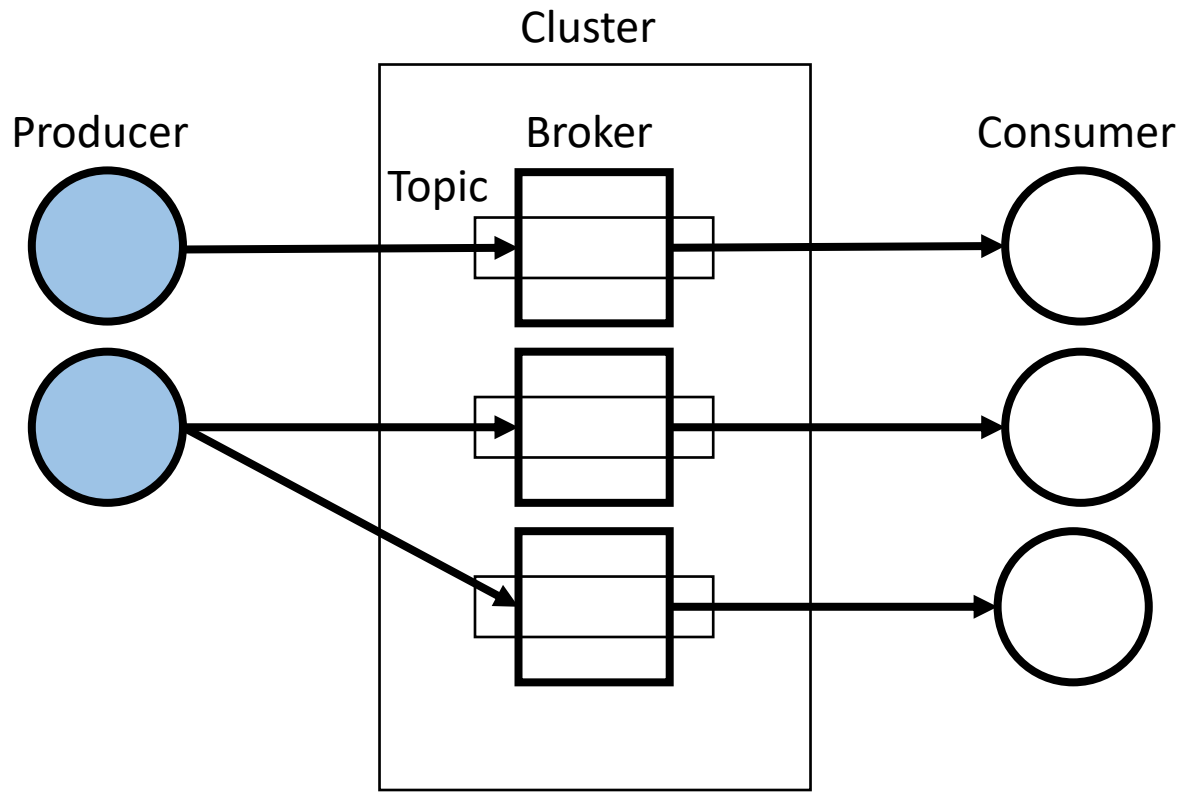
- 18 брокеров в Кластере
- 3 ДЦ (по 6 брокеров на 1 ДЦ)
- 3 млн сообщений / сек на кластер
- RF = 3
  
- 5 000 – 200 000 сообщений / сек на Producer
- 500 – 1 000 байтов на сообщение

# Apache Kafka



# Apache Kafka

## Kafka Producer



# Kafka Producer под капотом

# Kafka Producer под капотом

*Worker Thread*

```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```

# Kafka Producer под капотом

*Worker Thread*

```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```



# Kafka Producer под капотом

*Worker Thread*

```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```

# Kafka Producer под капотом

*Worker Thread*

```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```

max.block.ms

# Kafka Producer под капотом

*Worker Thread*

```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```

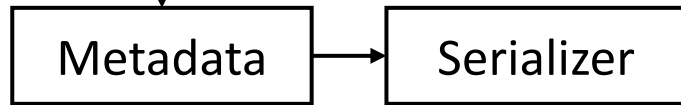
Metadata

max.block.ms

# Kafka Producer под капотом

*Worker Thread*

```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```



```
key.serializer  
value.serializer
```

# Kafka Producer под капотом

*Worker Thread*

```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```

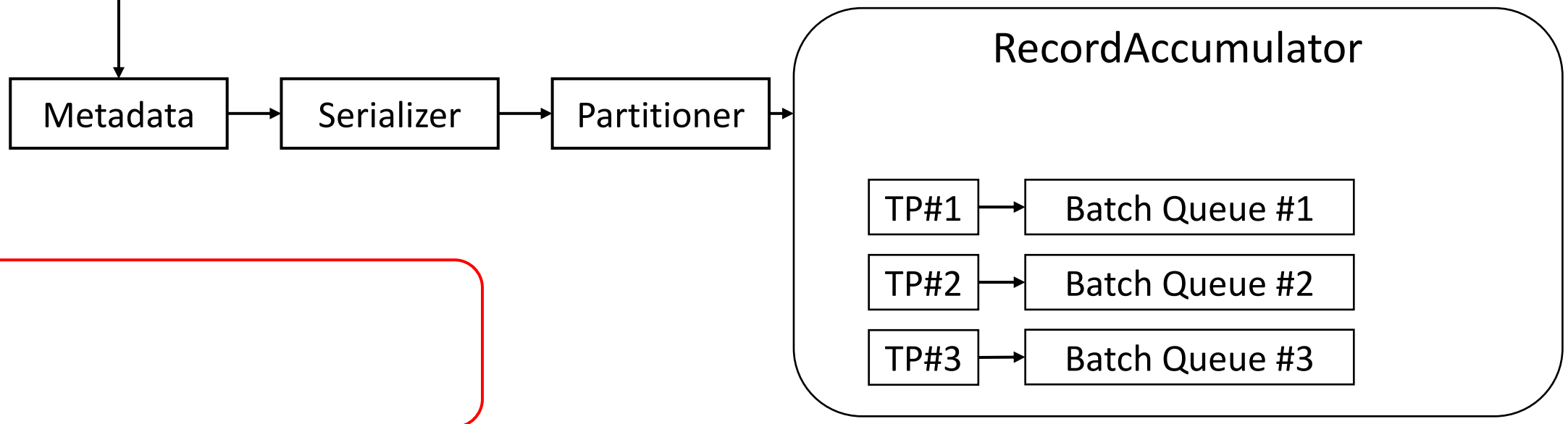


`partitioner.class`

# Kafka Producer под капотом

*Worker Thread*

```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```

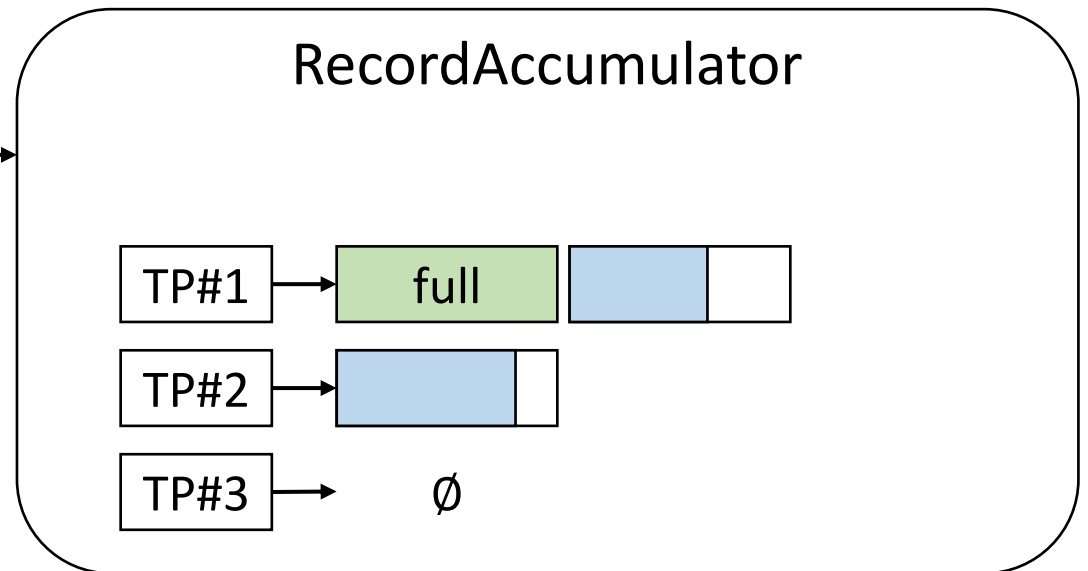


# Kafka Producer под капотом

*Worker Thread*

```
producer.send(
```

```
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```

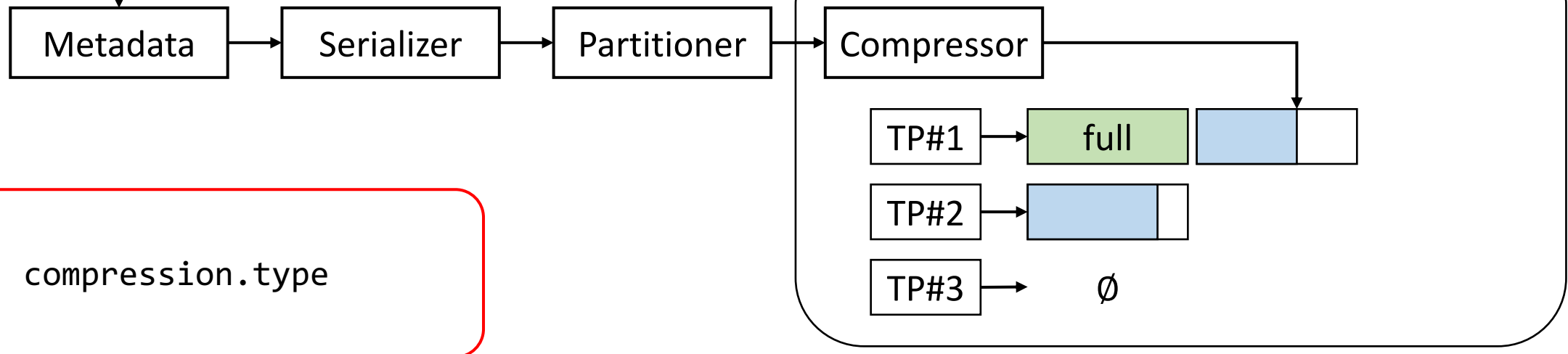


# Kafka Producer под капотом

*Worker Thread*

```
producer.send(
```

```
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```



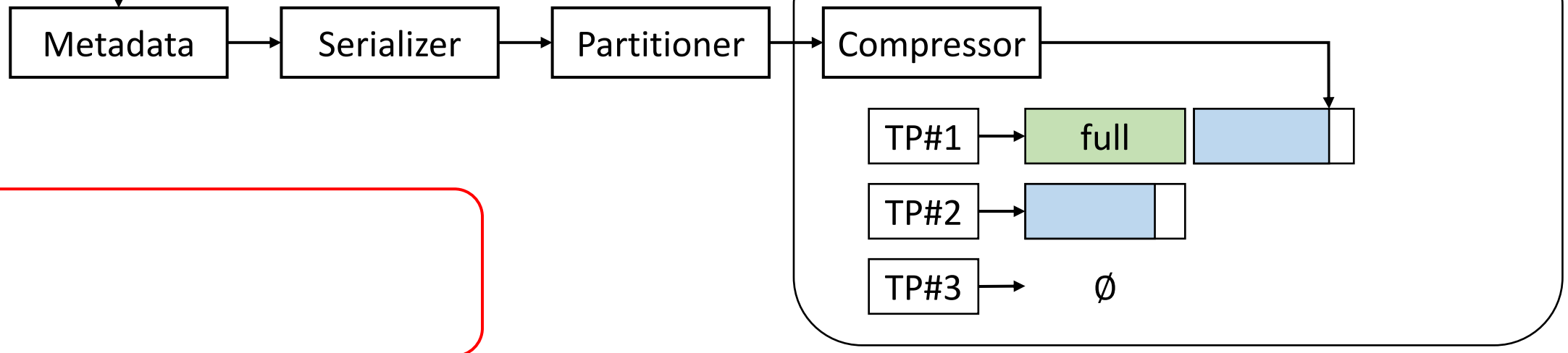


# Kafka Producer под капотом

*Worker Thread*

```
producer.send(
```

```
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```

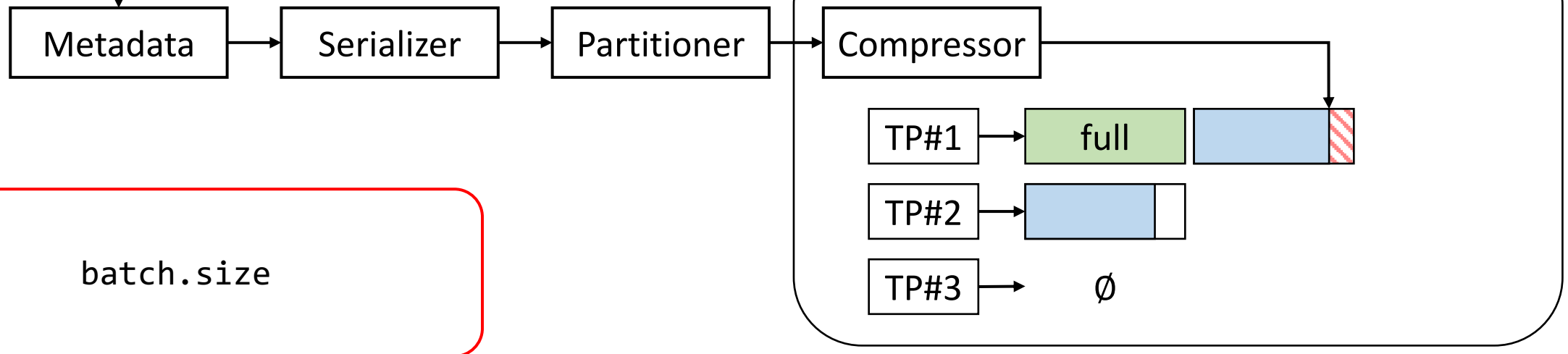


# Kafka Producer под капотом

*Worker Thread*

```
producer.send(
```

```
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```

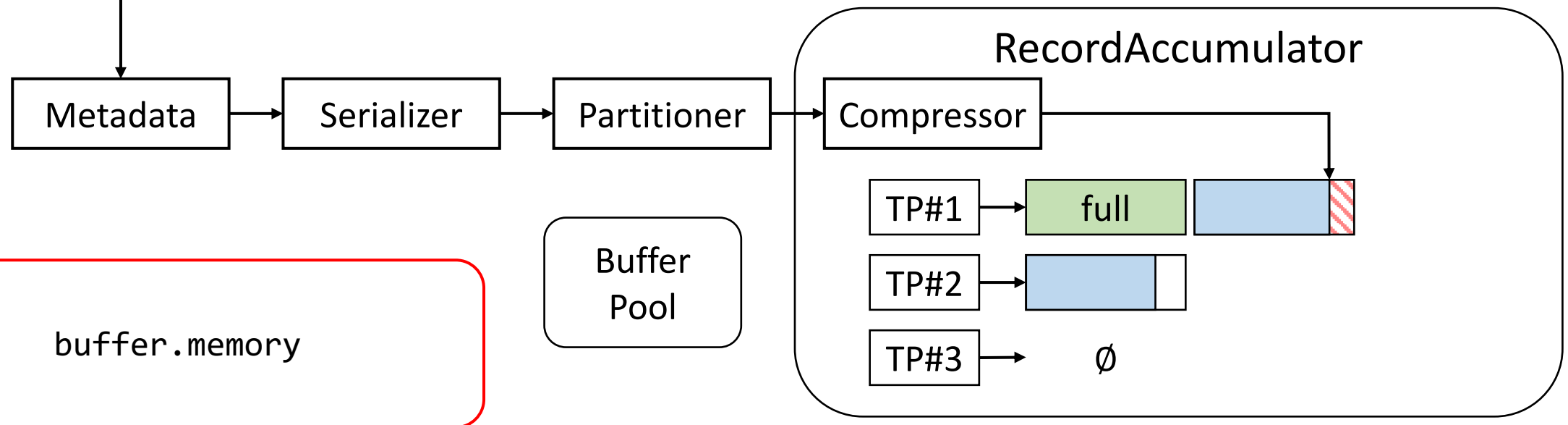


# Kafka Producer под капотом

*Worker Thread*

```
producer.send(
```

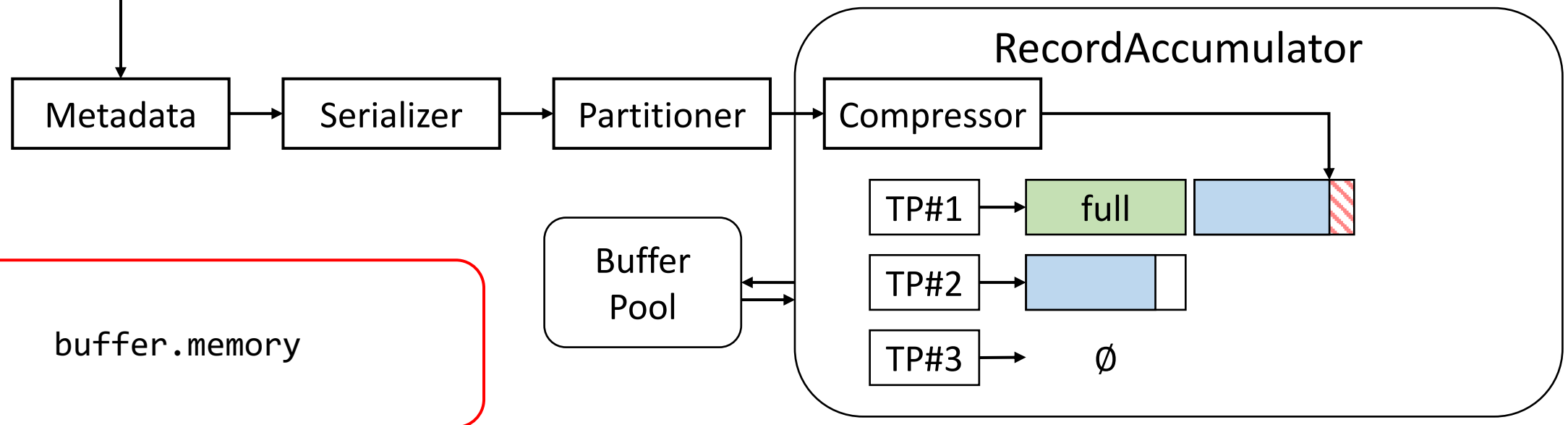
```
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```



# Kafka Producer под капотом

*Worker Thread*

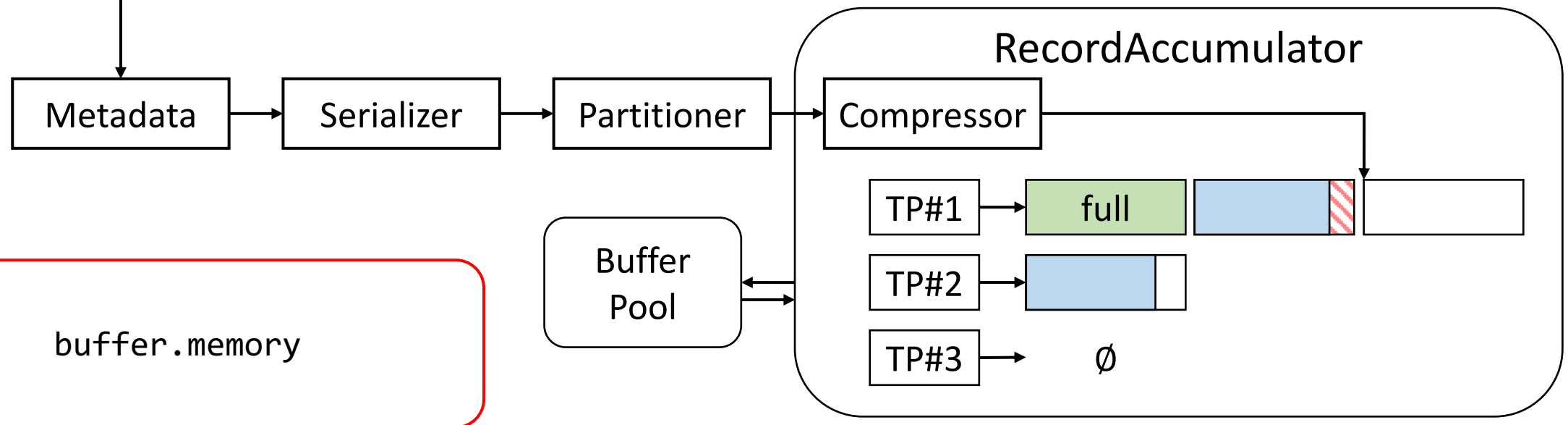
```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```



# Kafka Producer под капотом

*Worker Thread*

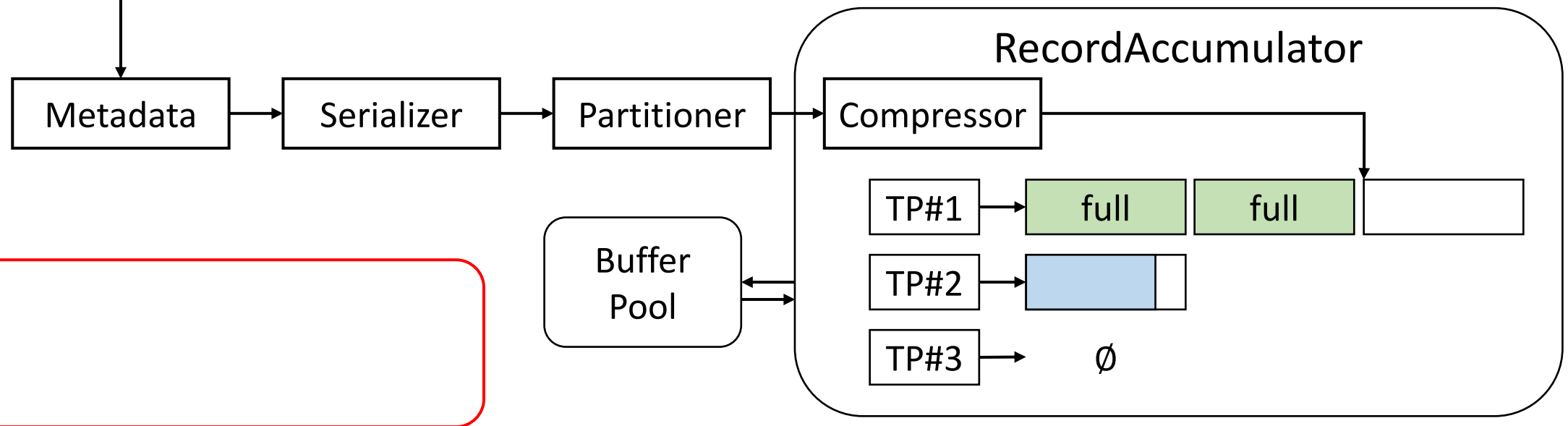
```
producer.send(  
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```



# Kafka Producer под капотом

*Worker Thread*

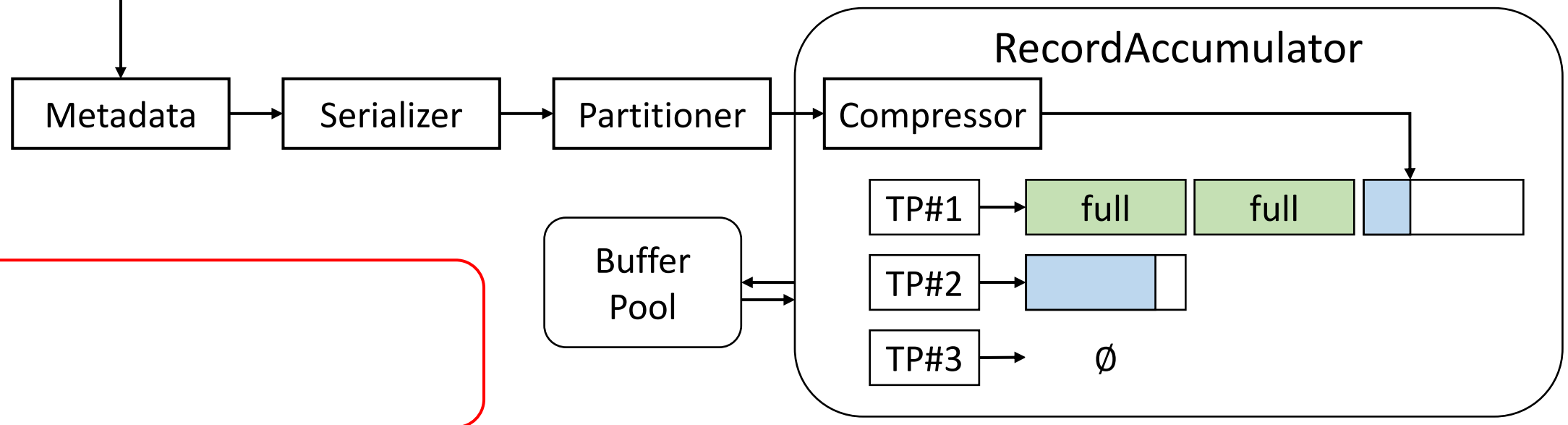
```
producer.send(  
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```



# Kafka Producer под капотом

*Worker Thread*

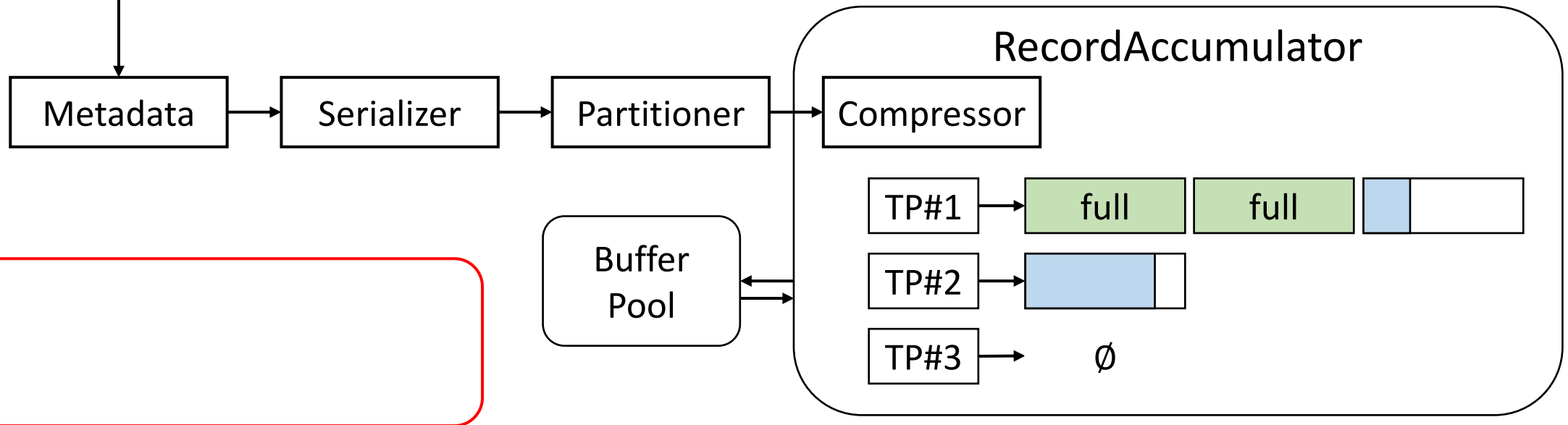
```
producer.send(  
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```



# Kafka Producer под капотом

*Worker Thread*

```
producer.send(  
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```

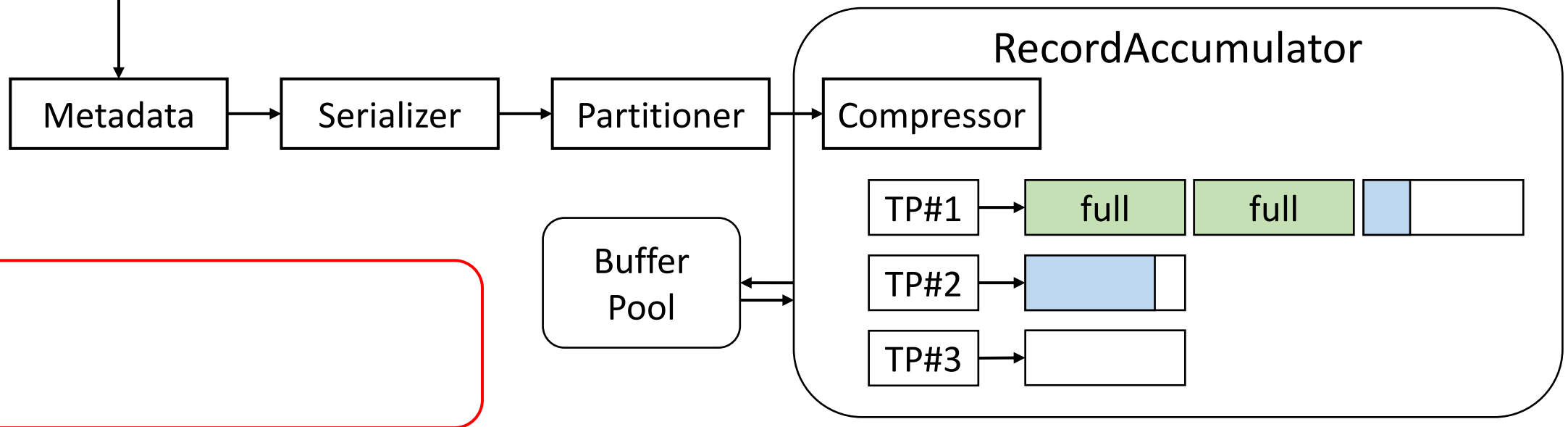




# Kafka Producer под капотом

*Worker Thread*

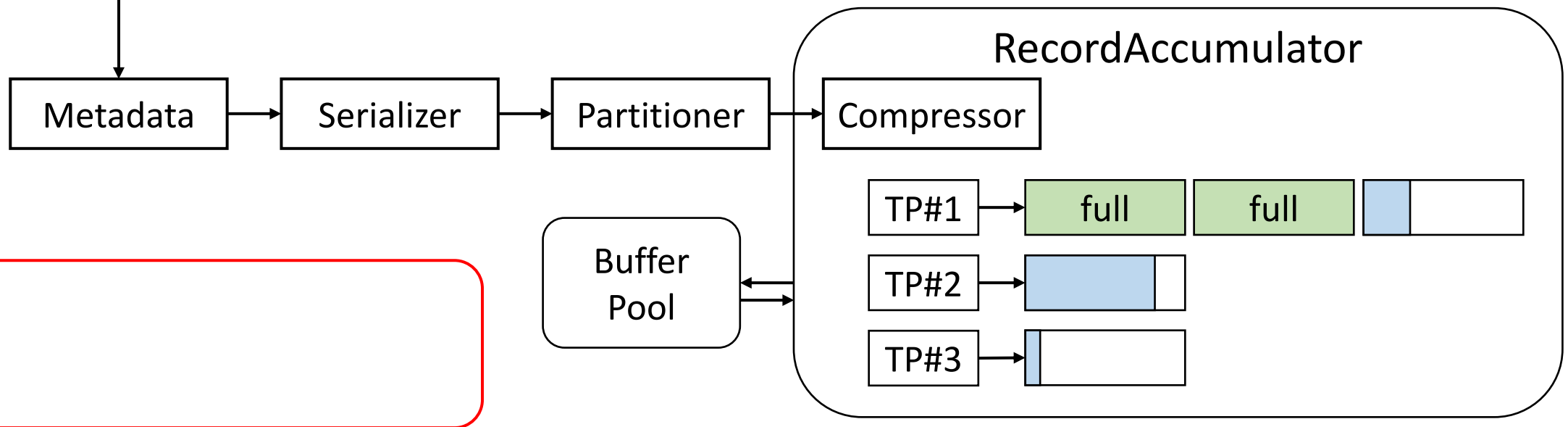
```
producer.send(  
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```



# Kafka Producer под капотом

*Worker Thread*

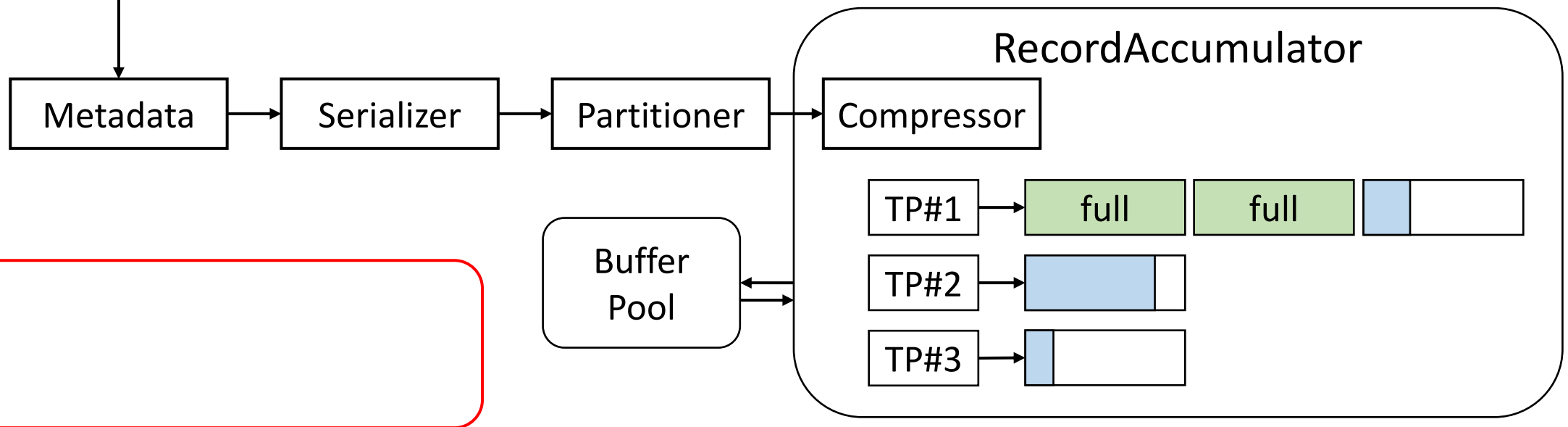
```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```



# Kafka Producer под капотом

*Worker Thread*

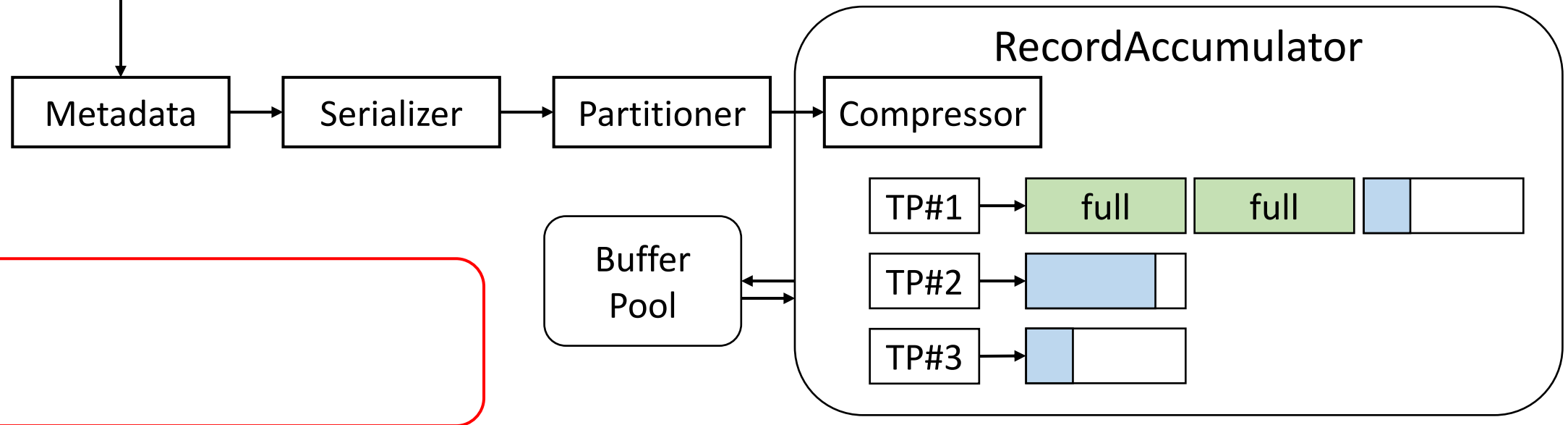
```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```



# Kafka Producer под капотом

*Worker Thread*

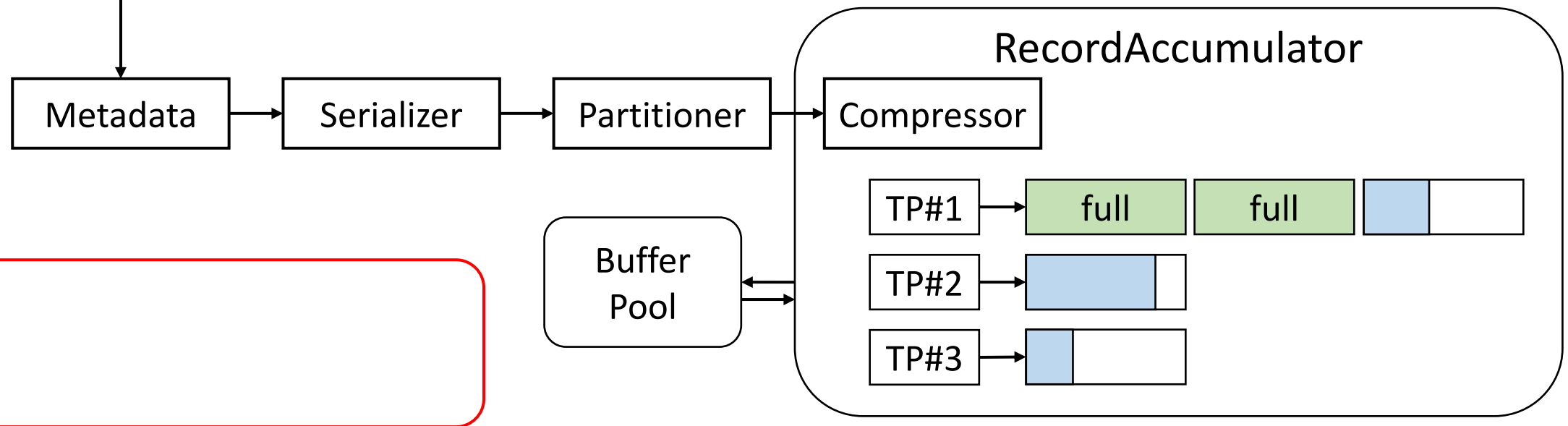
```
producer.send(  
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```



# Kafka Producer под капотом

*Worker Thread*

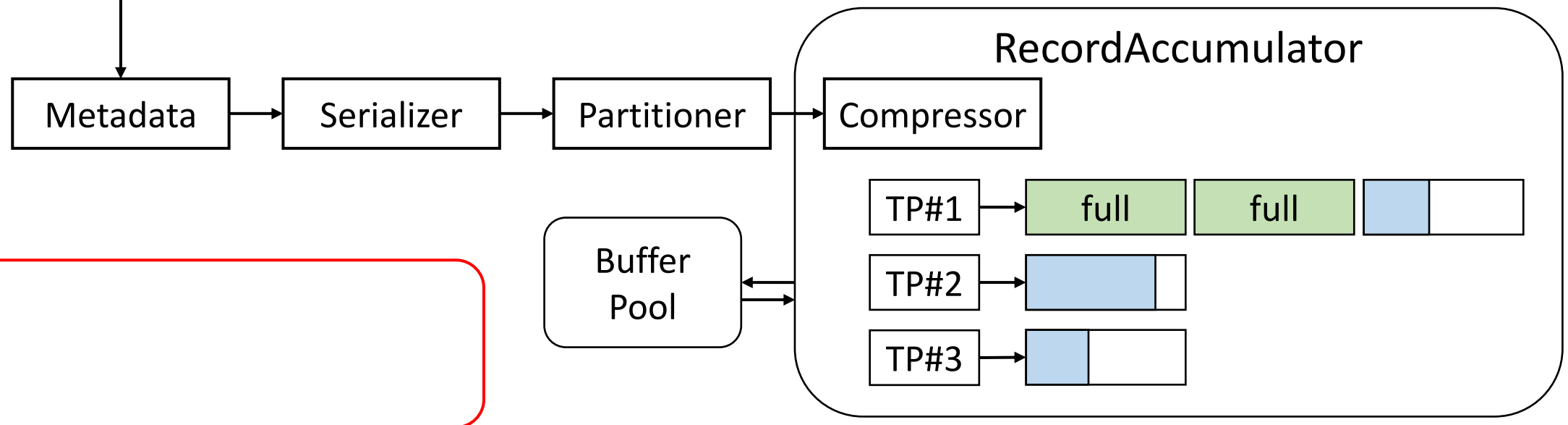
```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```



# Kafka Producer под капотом

*Worker Thread*

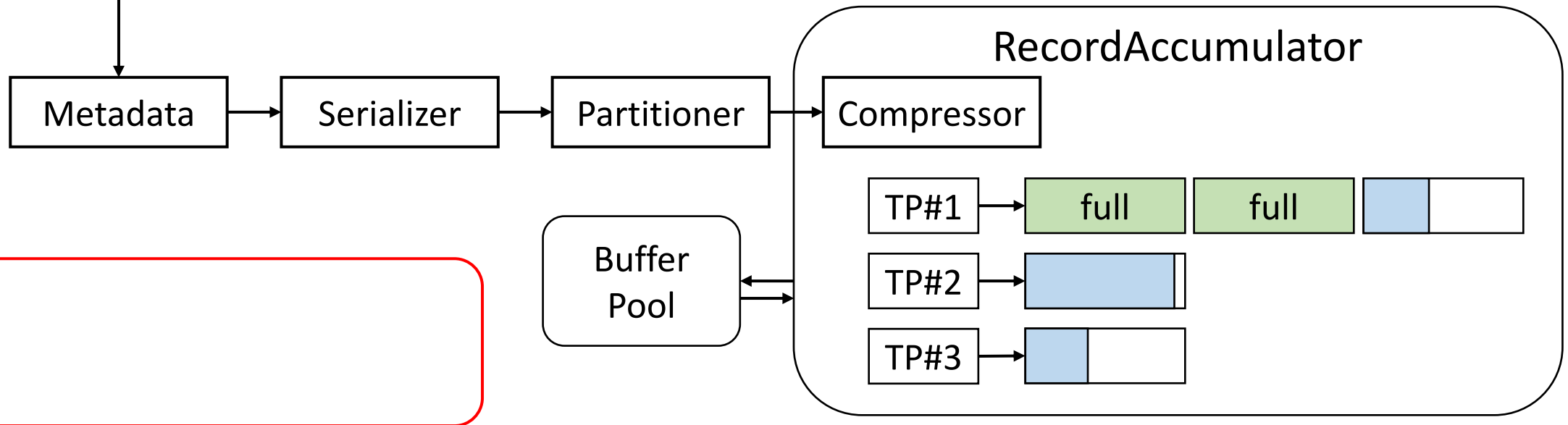
```
producer.send(  
  new ProducerRecord<>(topic, partition, key, value),  
  callback);
```



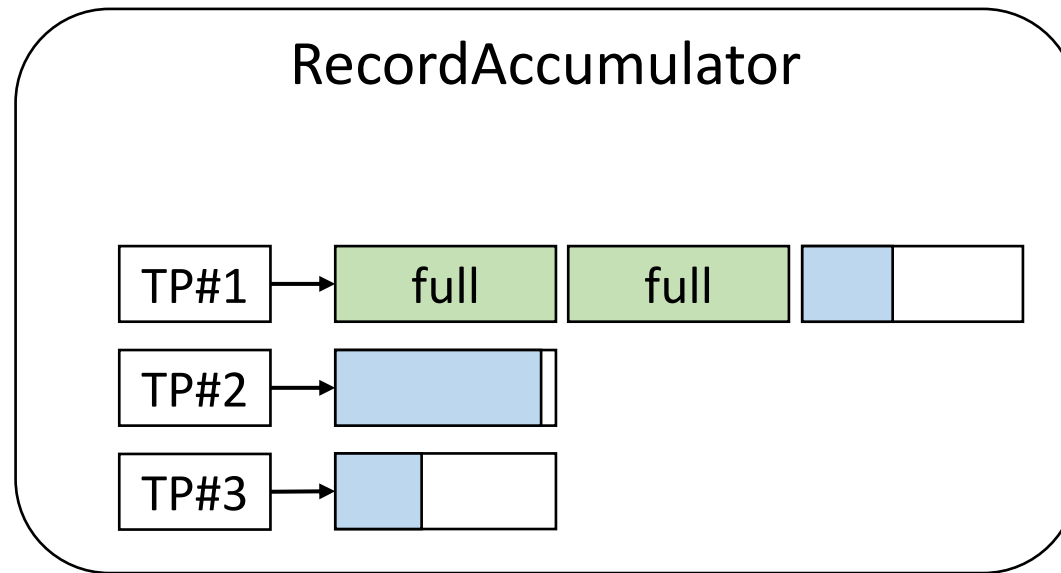
# Kafka Producer под капотом

*Worker Thread*

```
producer.send(  
    new ProducerRecord<>(topic, partition, key, value),  
    callback);
```



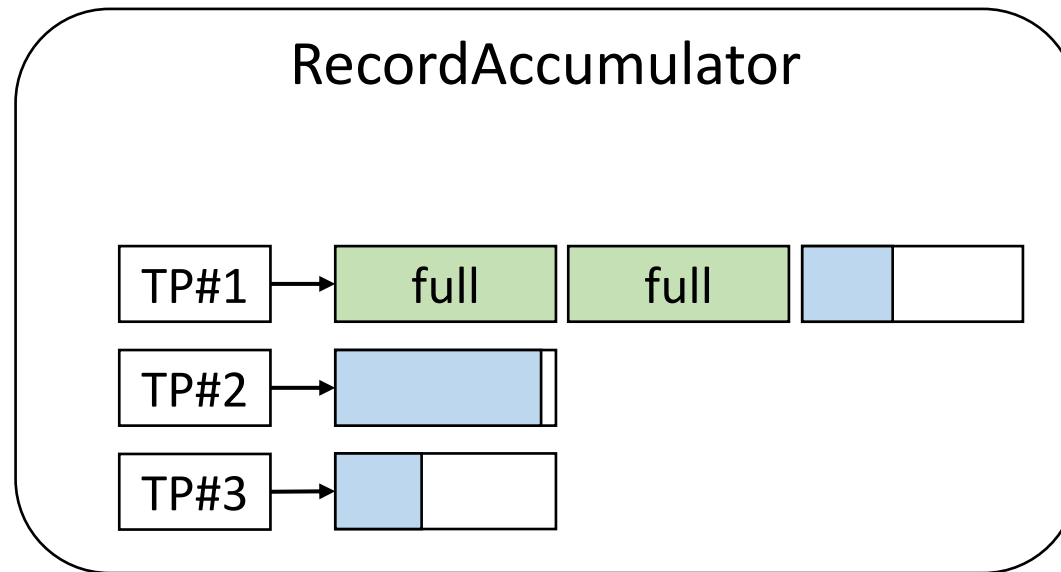
# Kafka Producer под капотом



*Sender Thread*



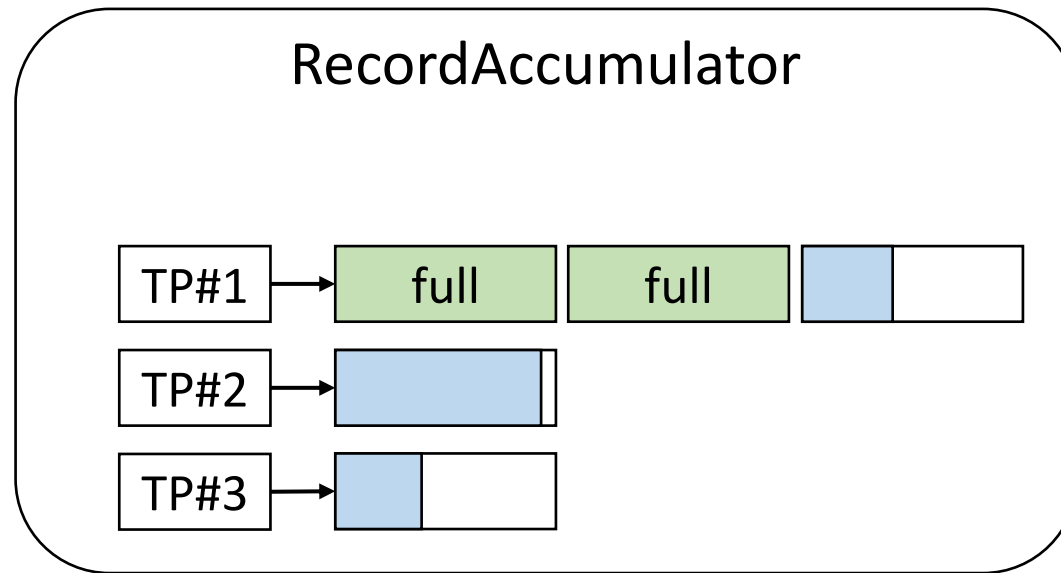
# Kafka Producer под капотом



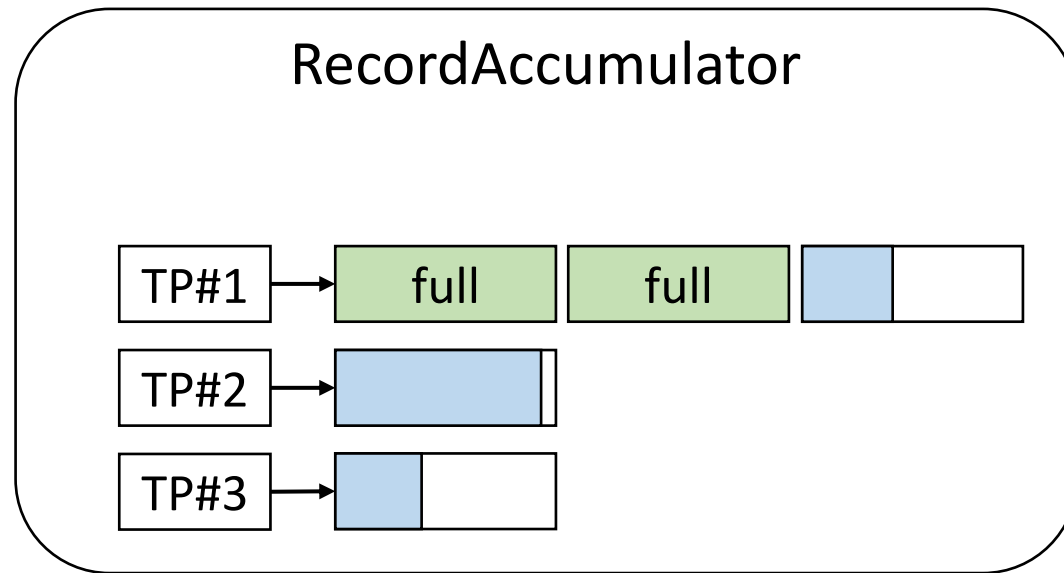
*Sender Thread*

# Kafka Producer под капотом

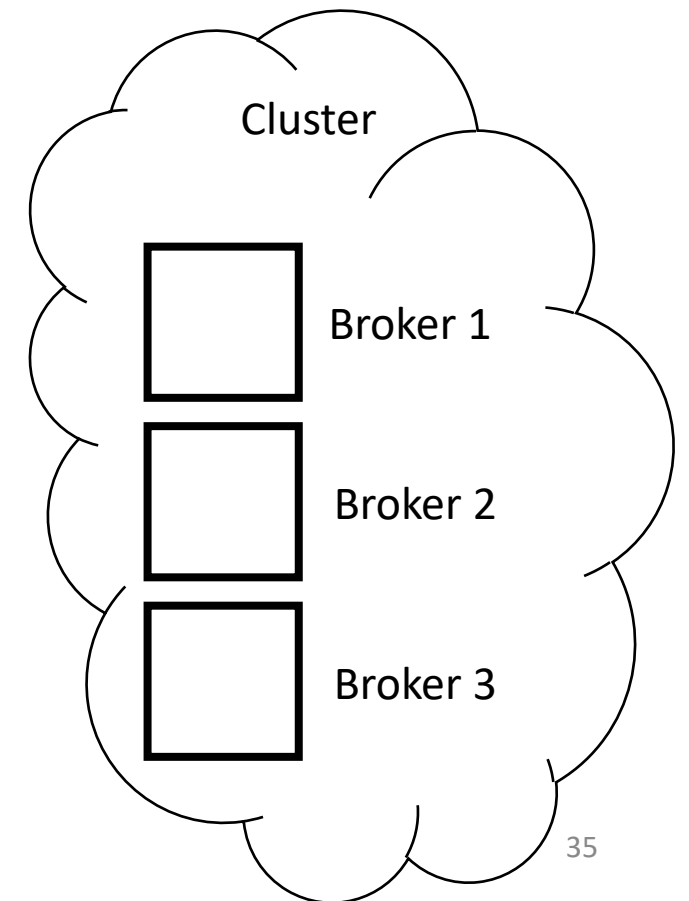
*Sender Thread*



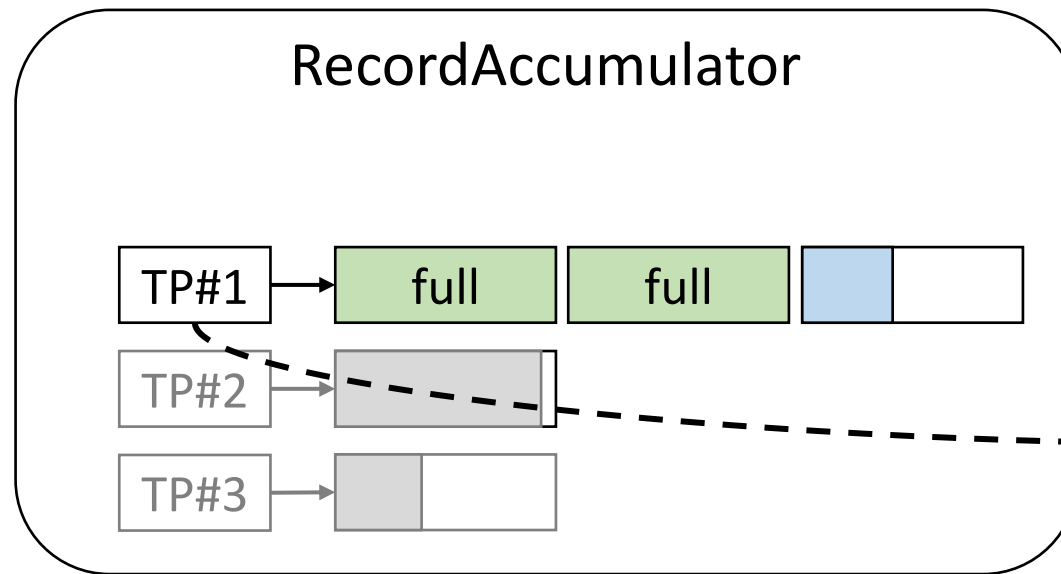
# Kafka Producer под капотом



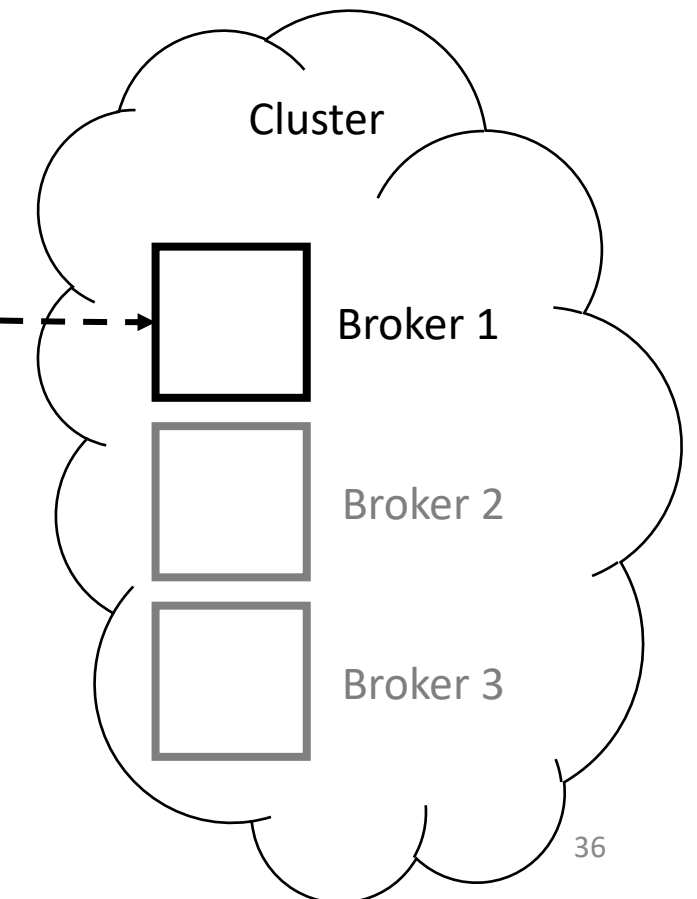
## *Sender Thread*



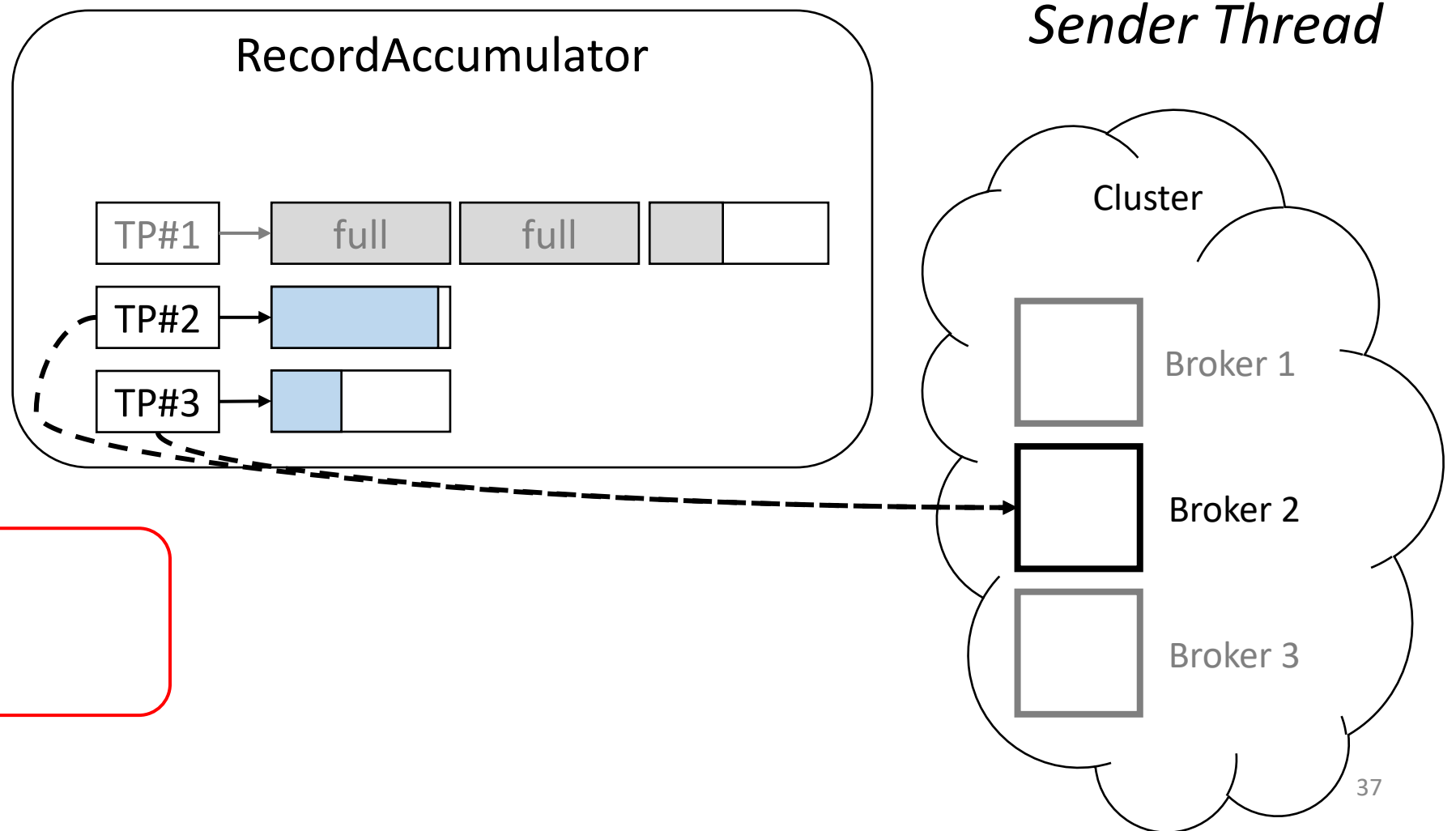
# Kafka Producer под капотом



*Sender Thread*

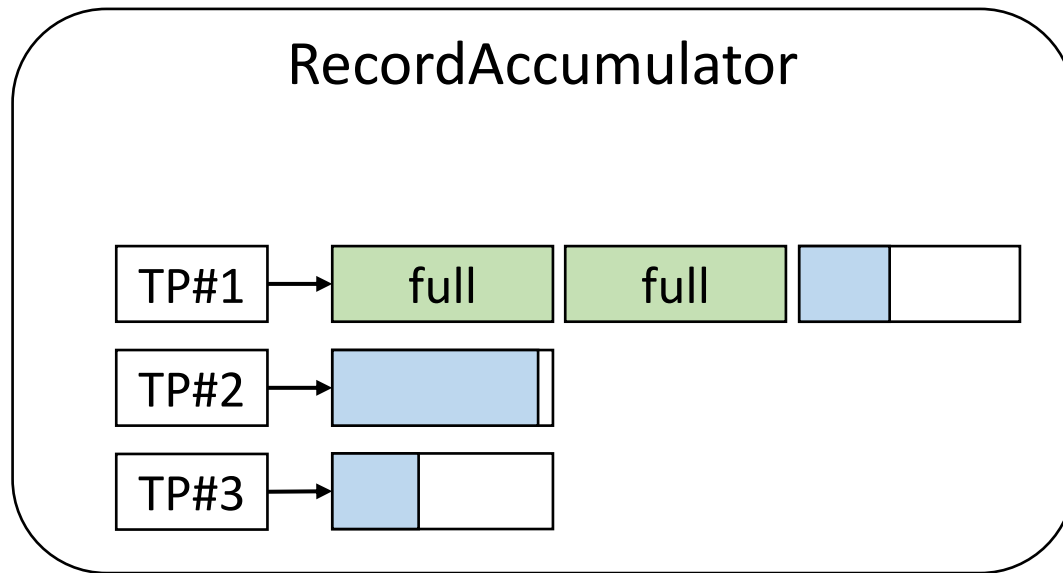


# Kafka Producer под капотом

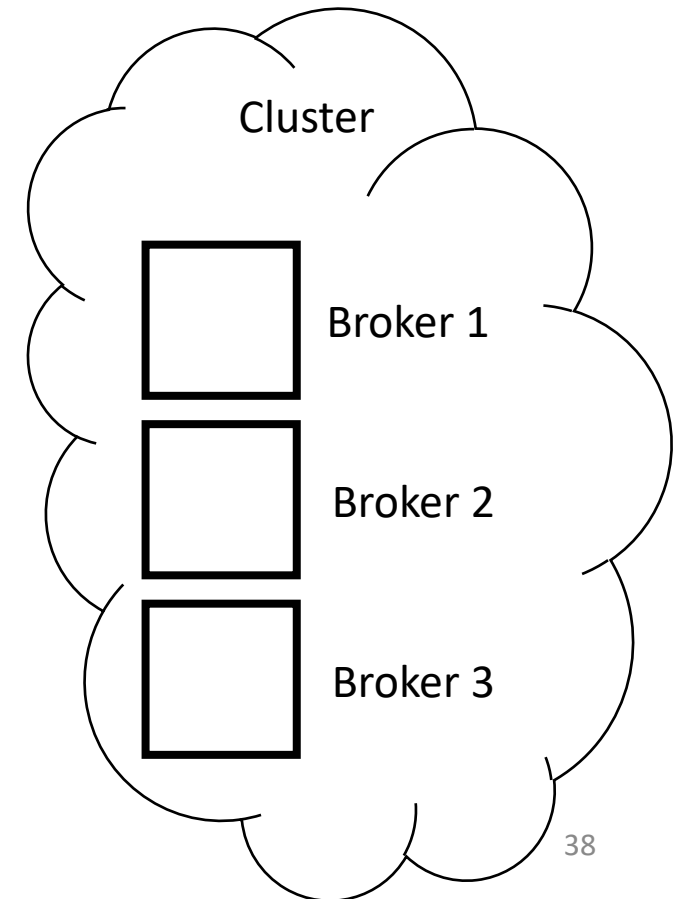


# Kafka Producer под капотом

Drain batches

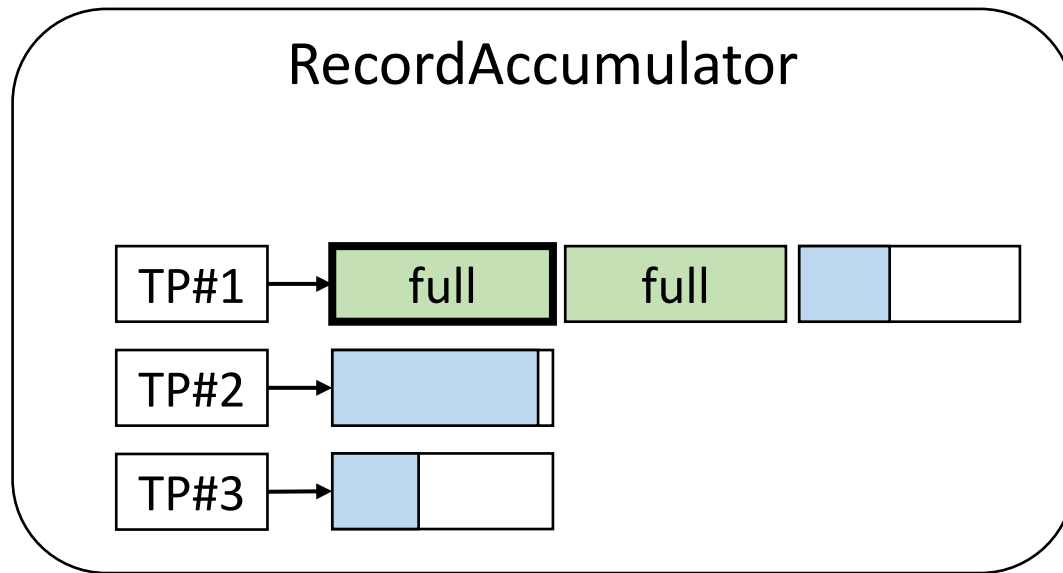


*Sender Thread*

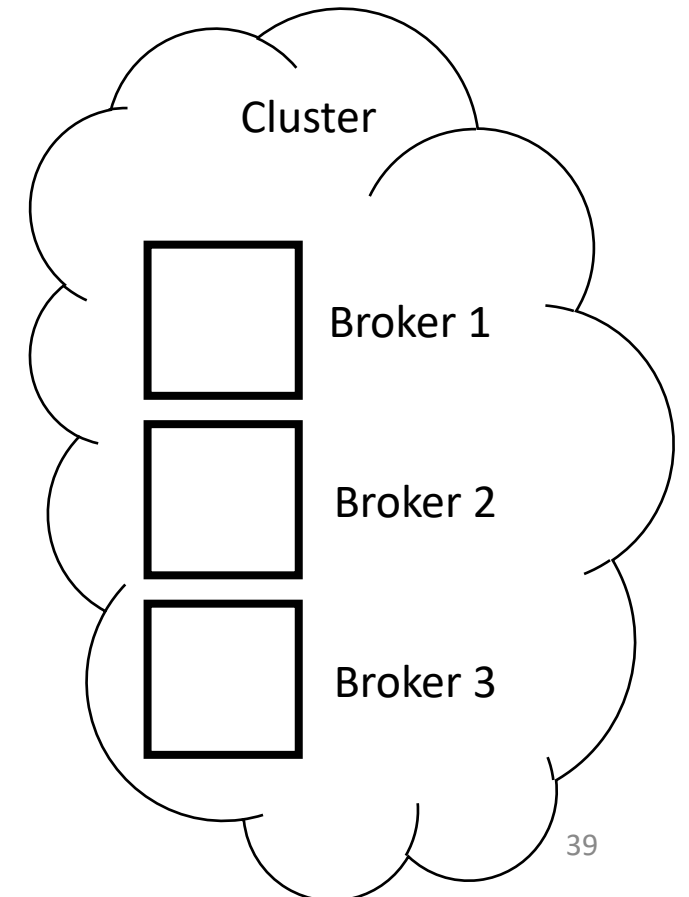


# Kafka Producer под капотом

Drain batches



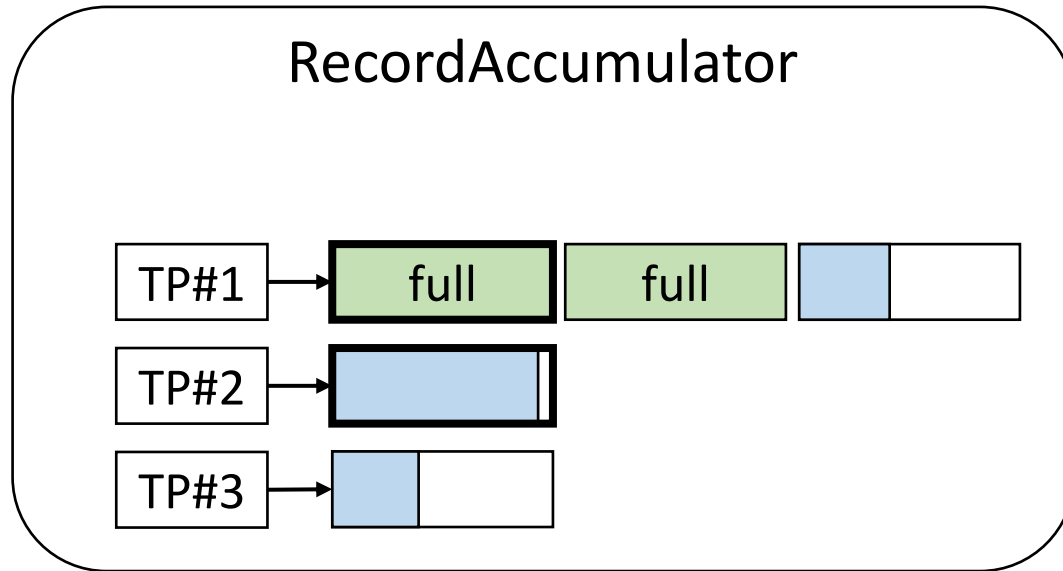
*Sender Thread*



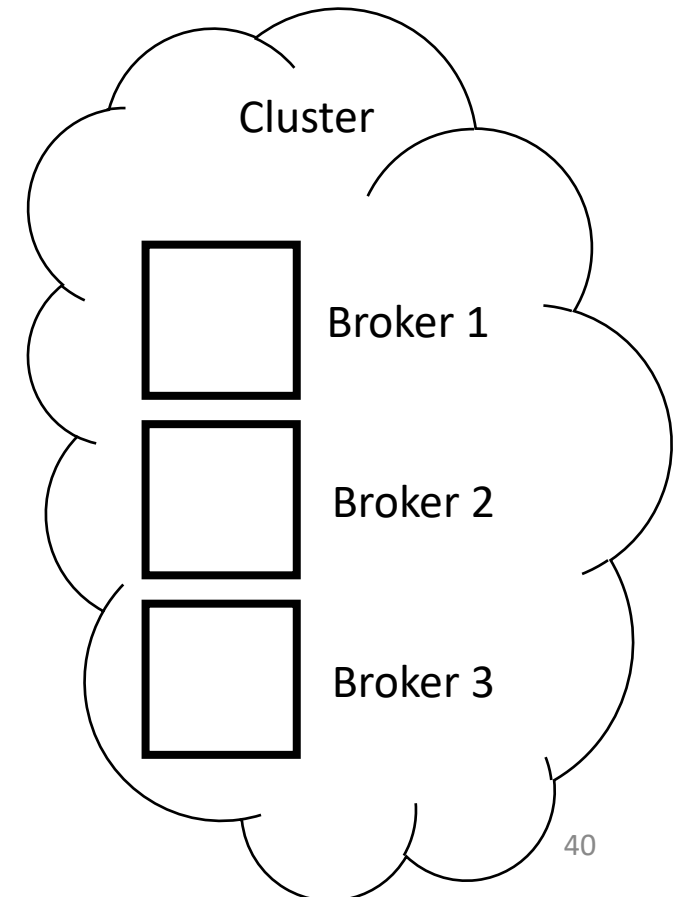
batch.size

# Kafka Producer под капотом

Drain batches



*Sender Thread*

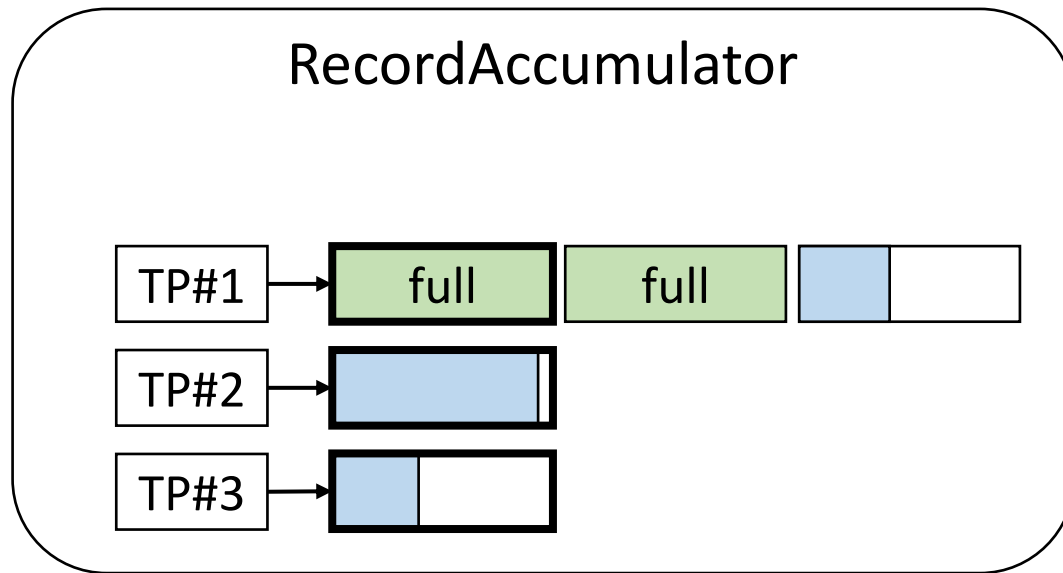


`linger.ms`

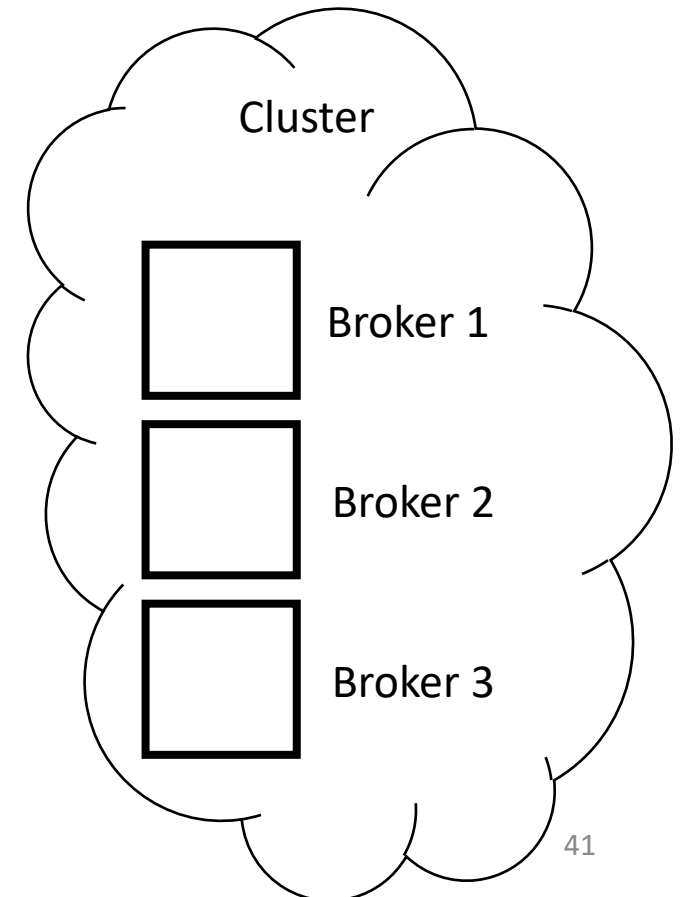


# Kafka Producer под капотом

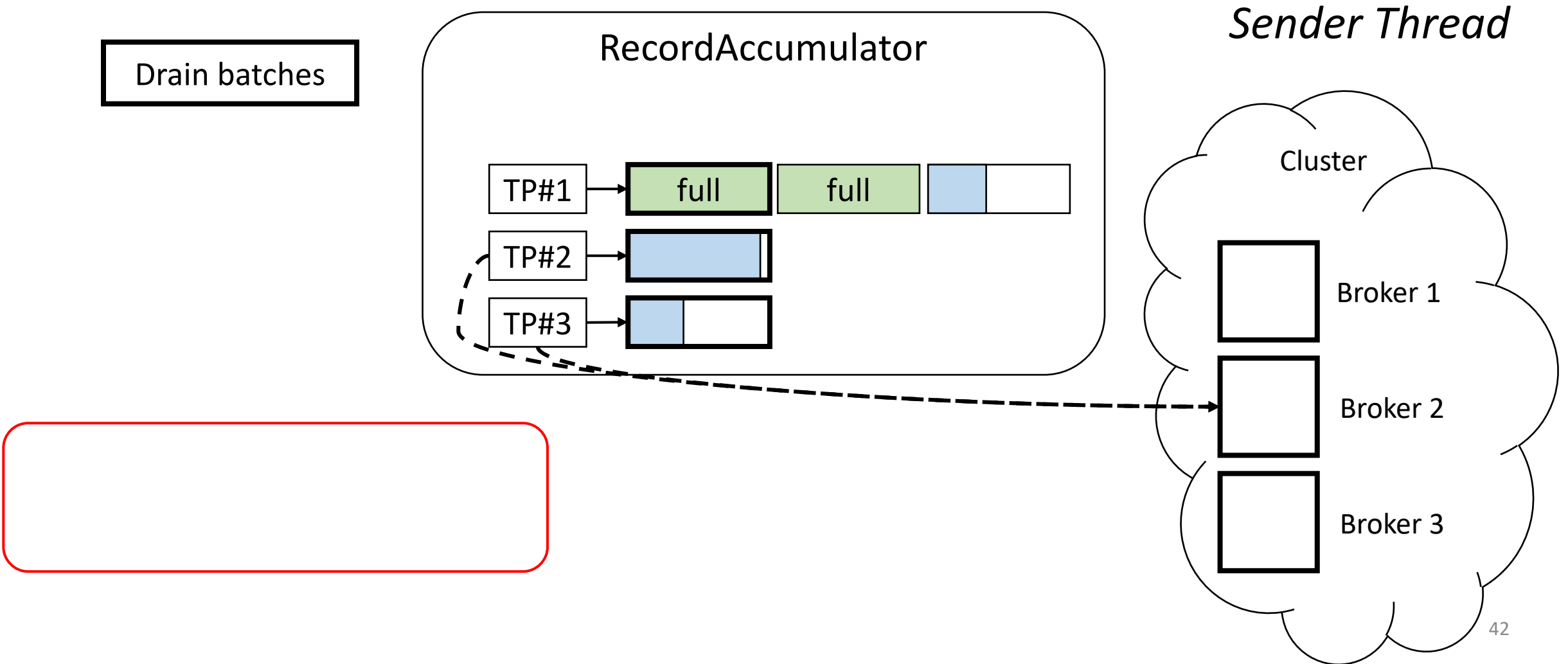
Drain batches



*Sender Thread*

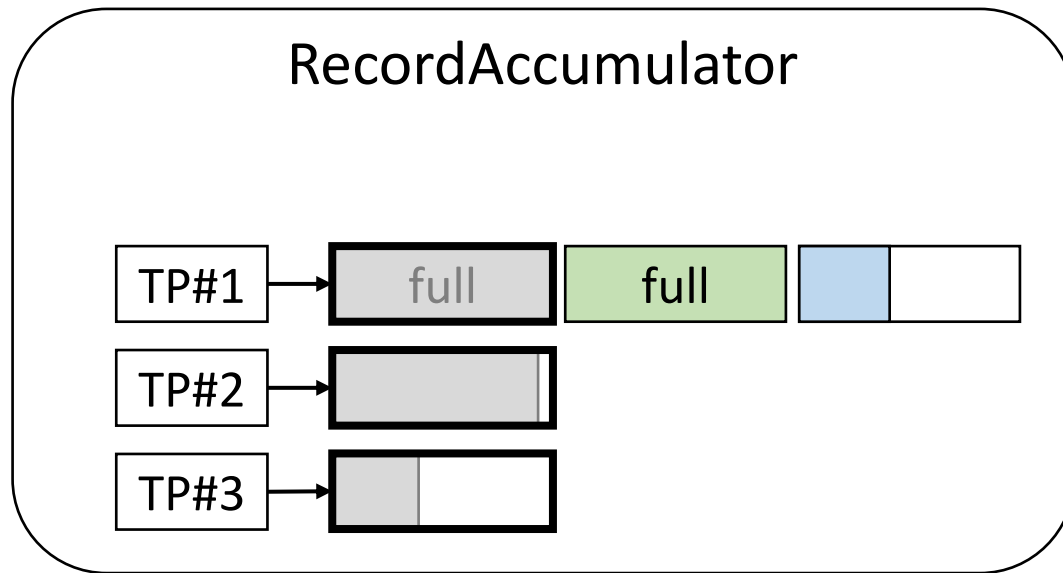


# Kafka Producer под капотом

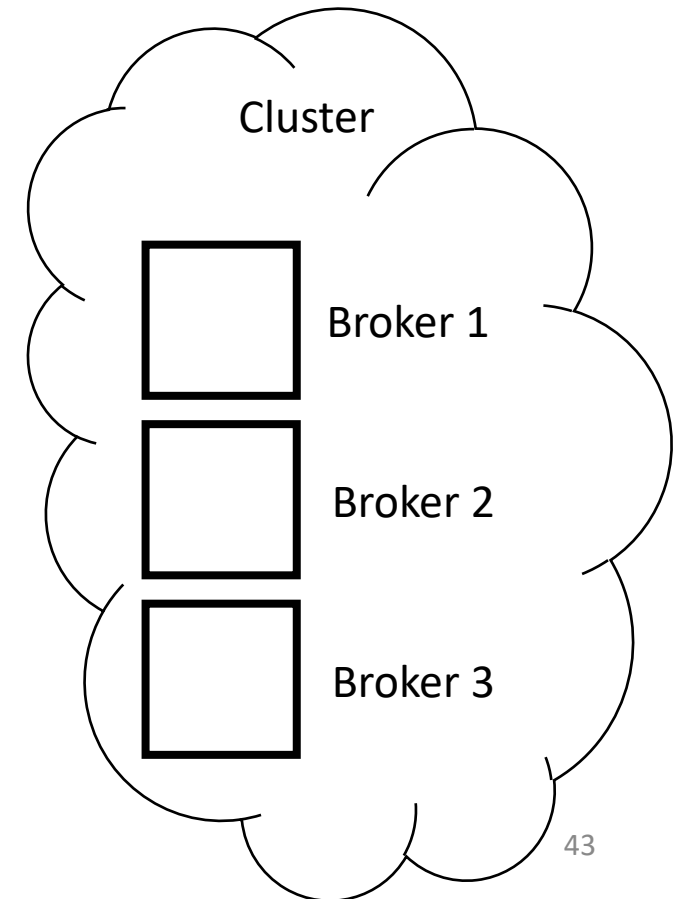


# Kafka Producer под капотом

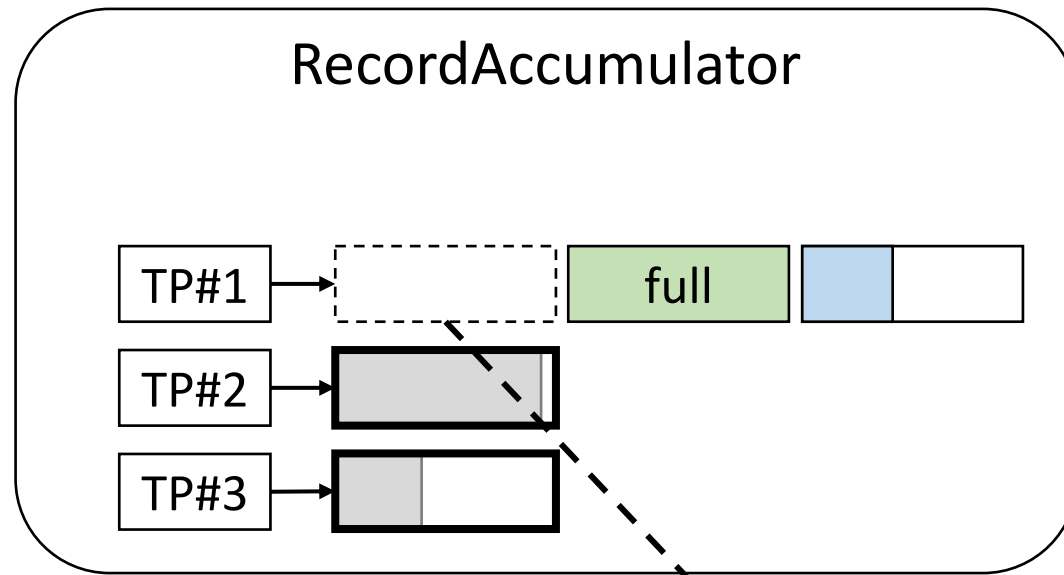
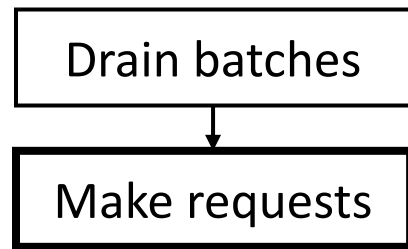
Drain batches



*Sender Thread*



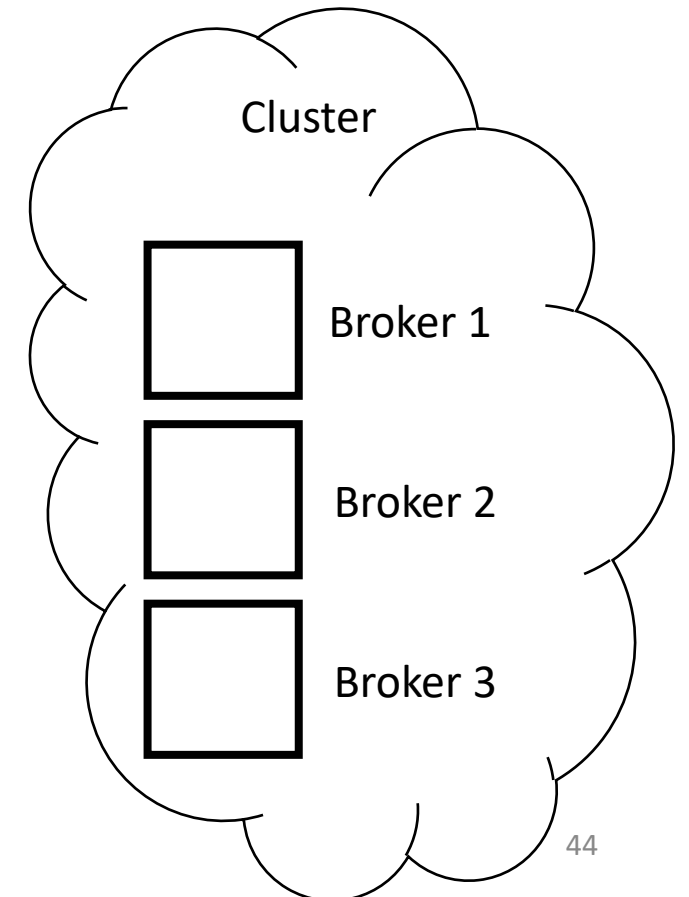
# Kafka Producer под капотом



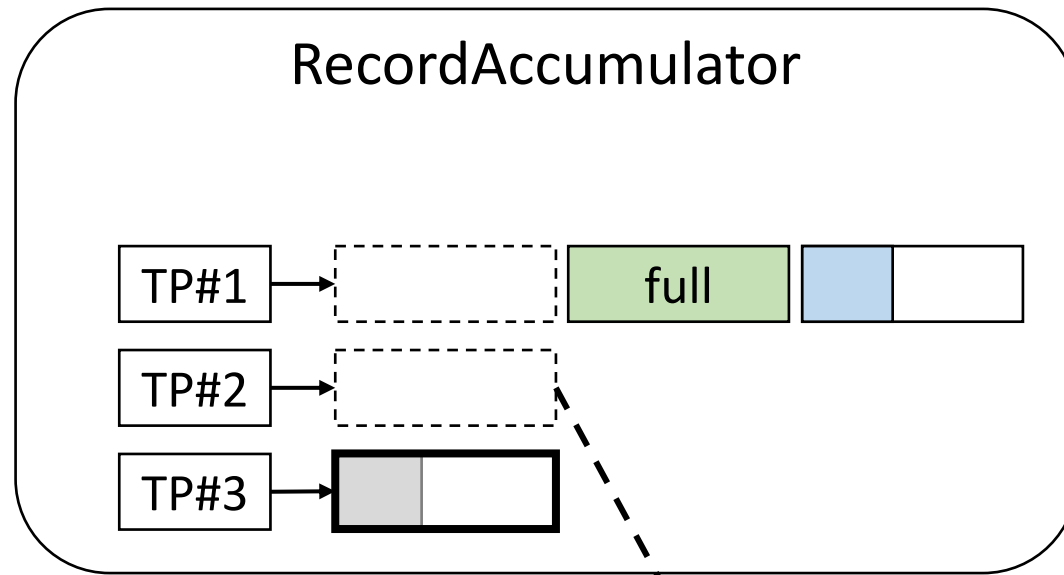
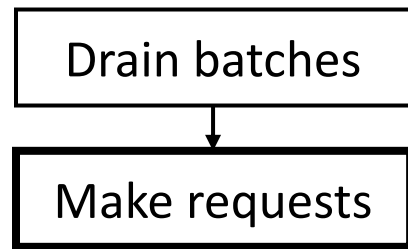
ProduceRequest#1 [ full ]

`max.request.size`

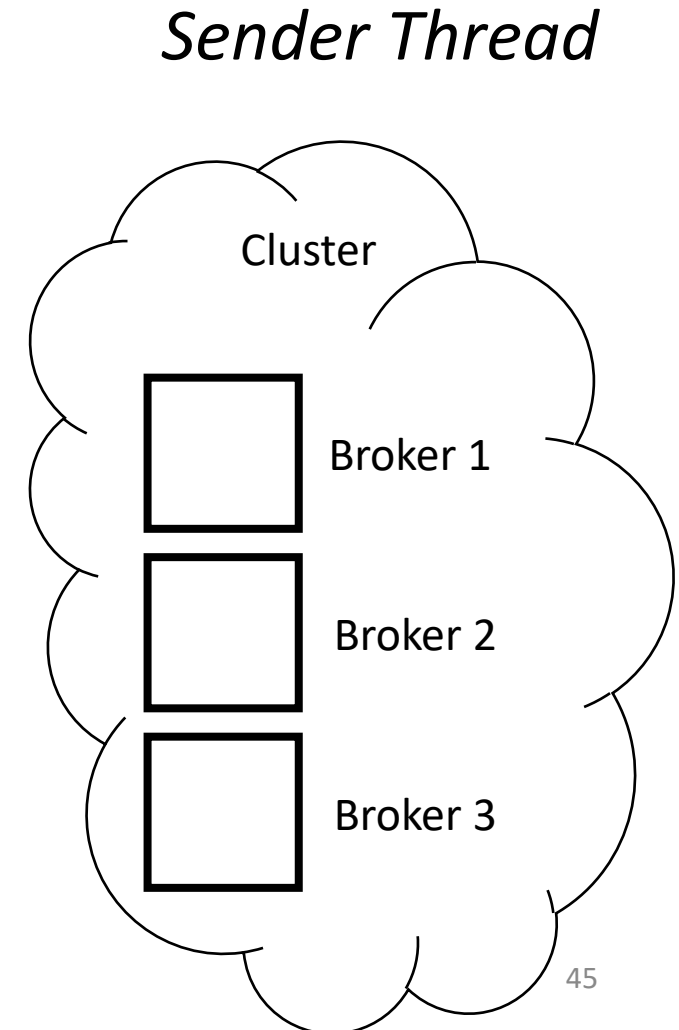
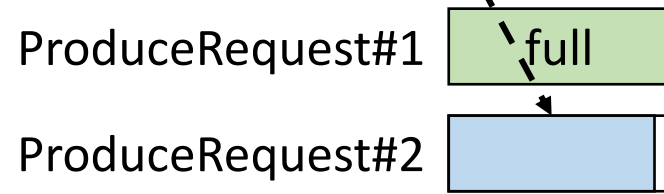
## *Sender Thread*



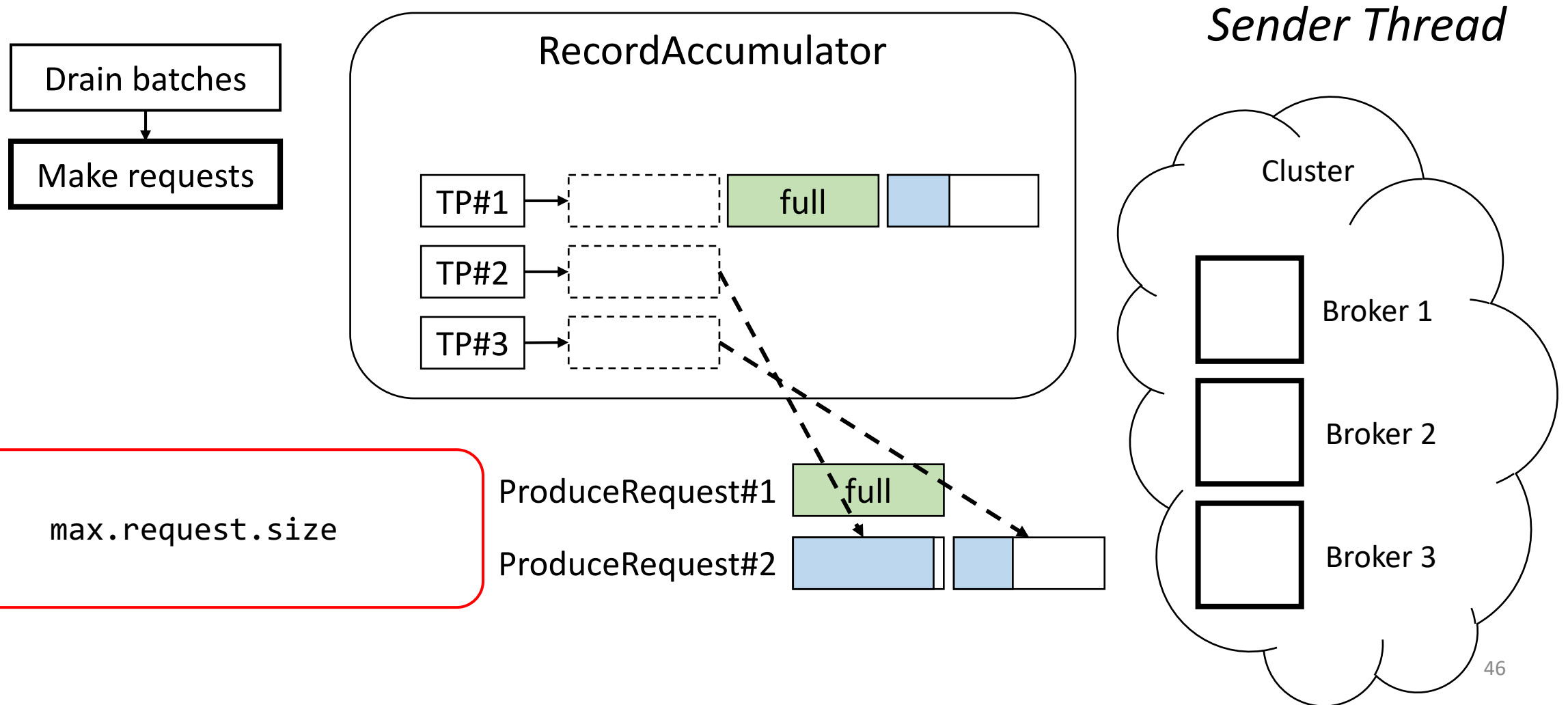
# Kafka Producer под капотом



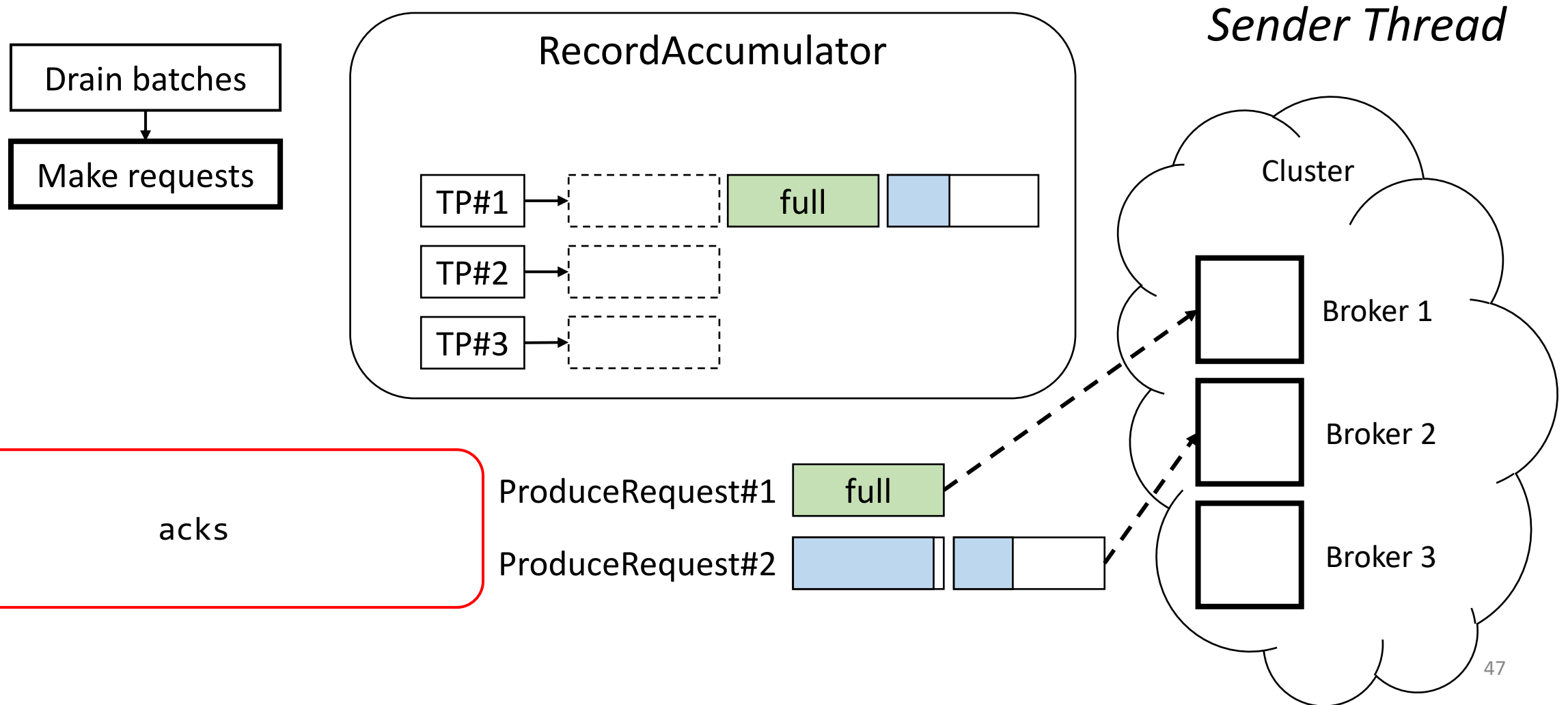
max.request.size



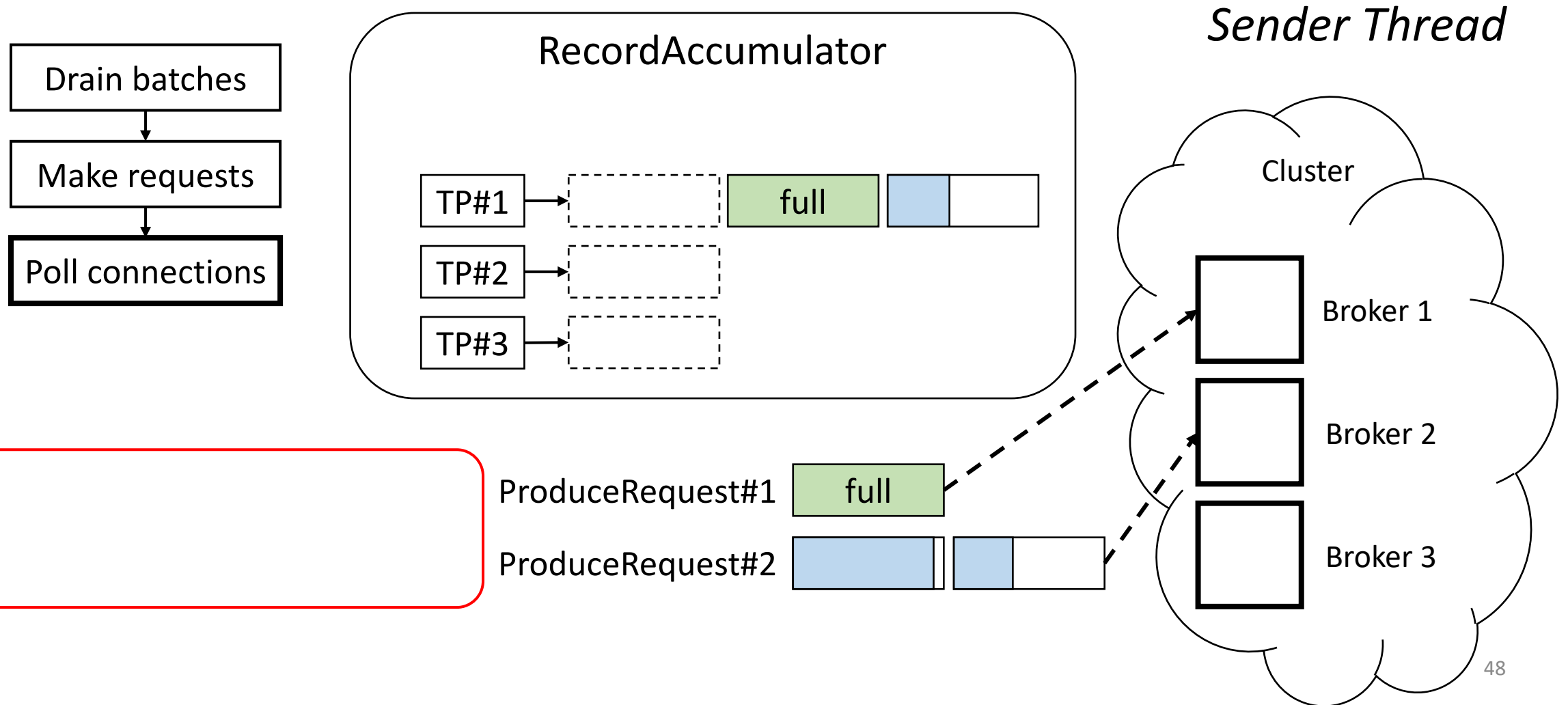
# Kafka Producer под капотом



# Kafka Producer под капотом

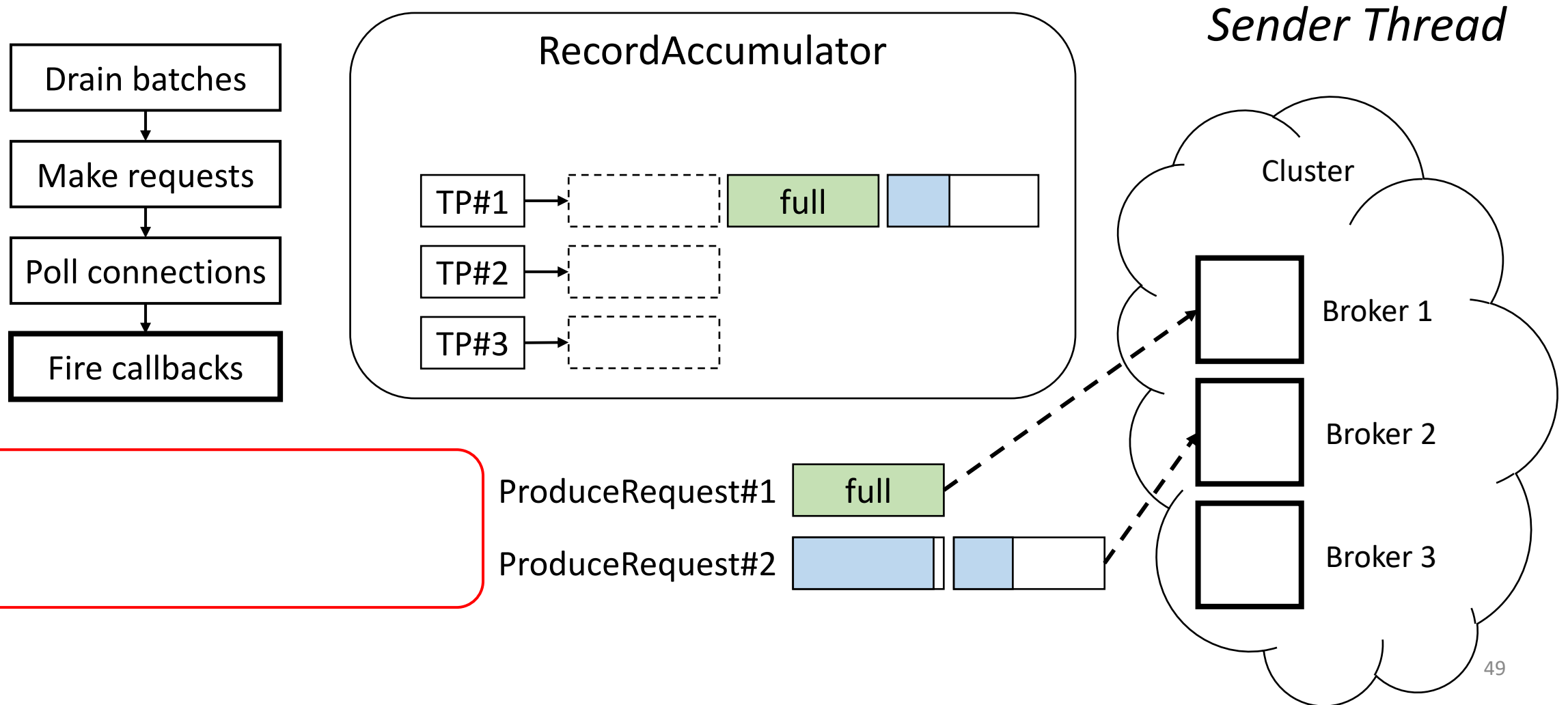


# Kafka Producer под капотом

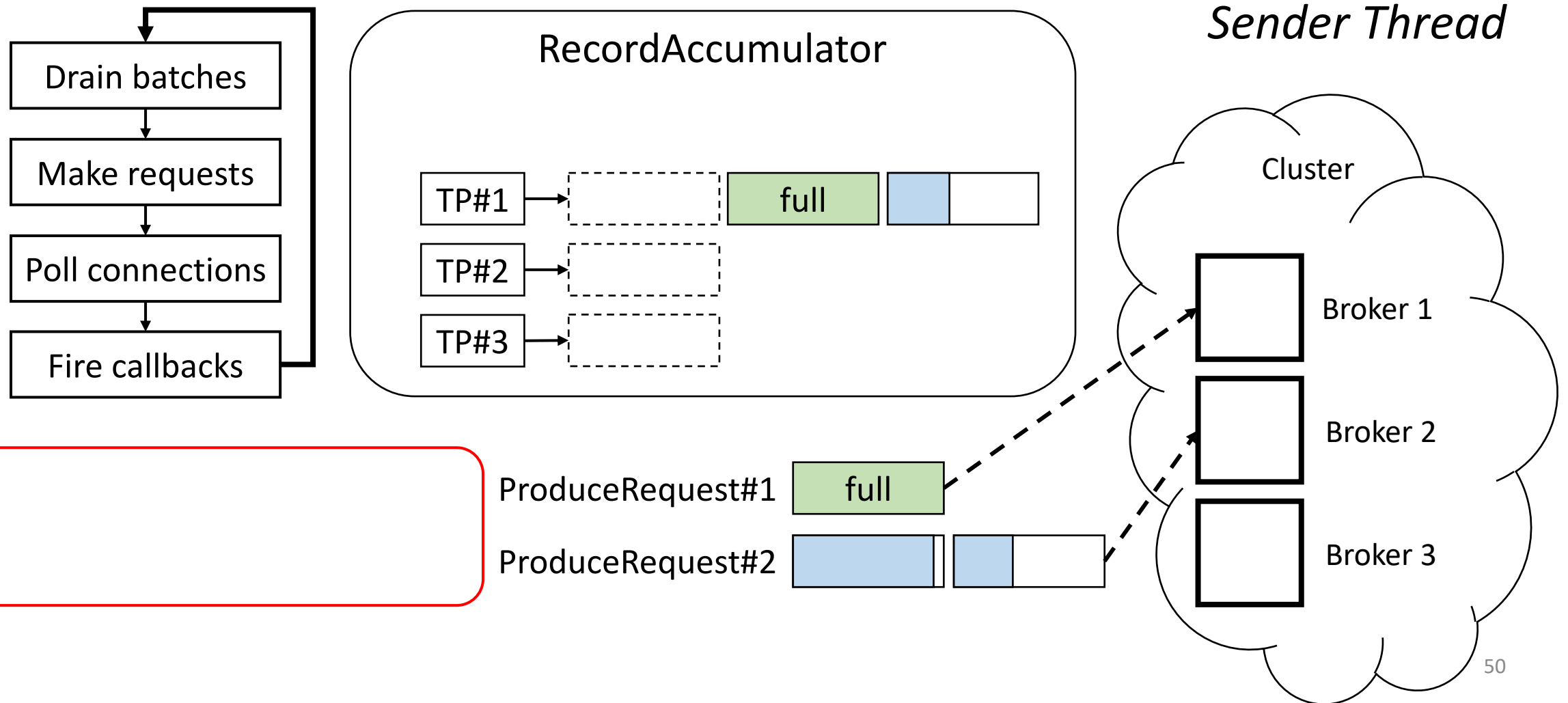




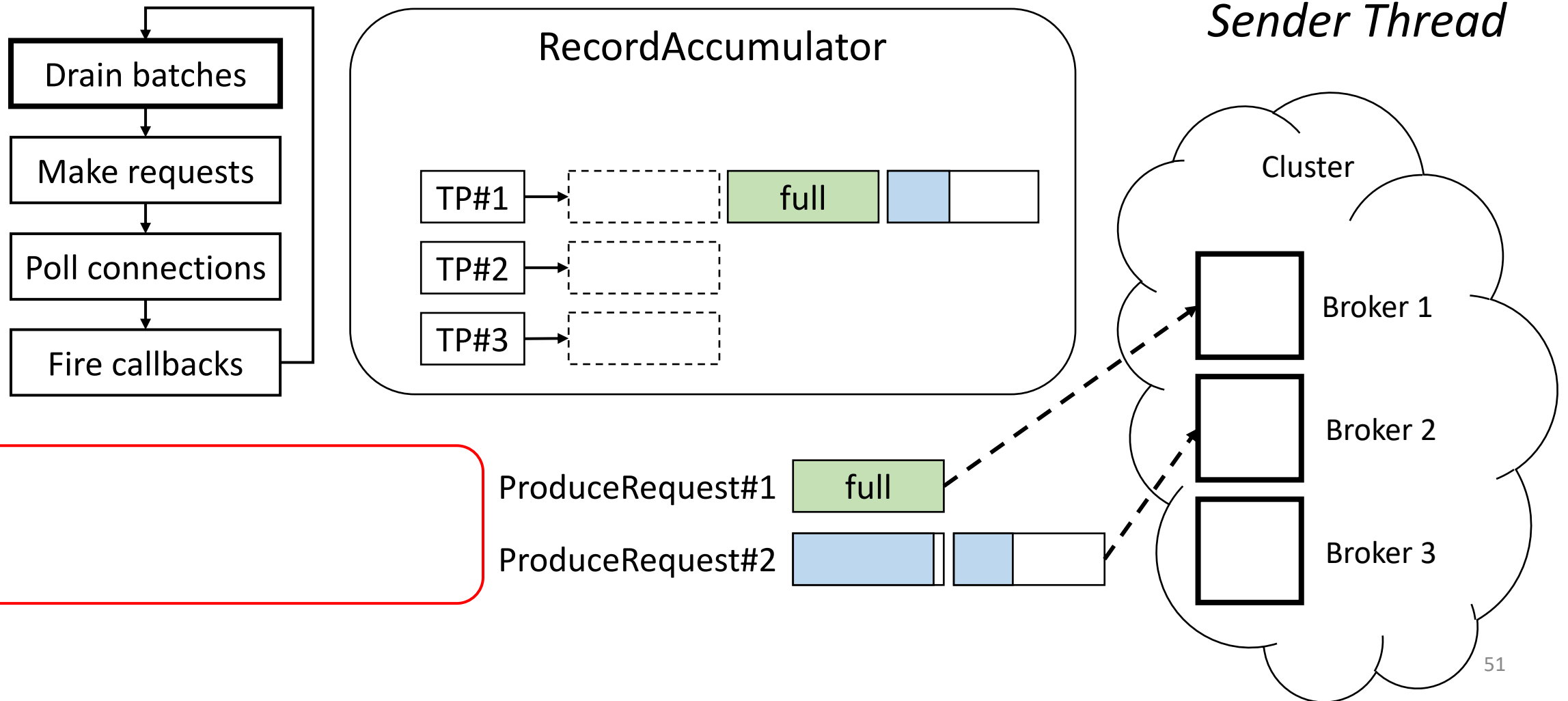
# Kafka Producer под капотом



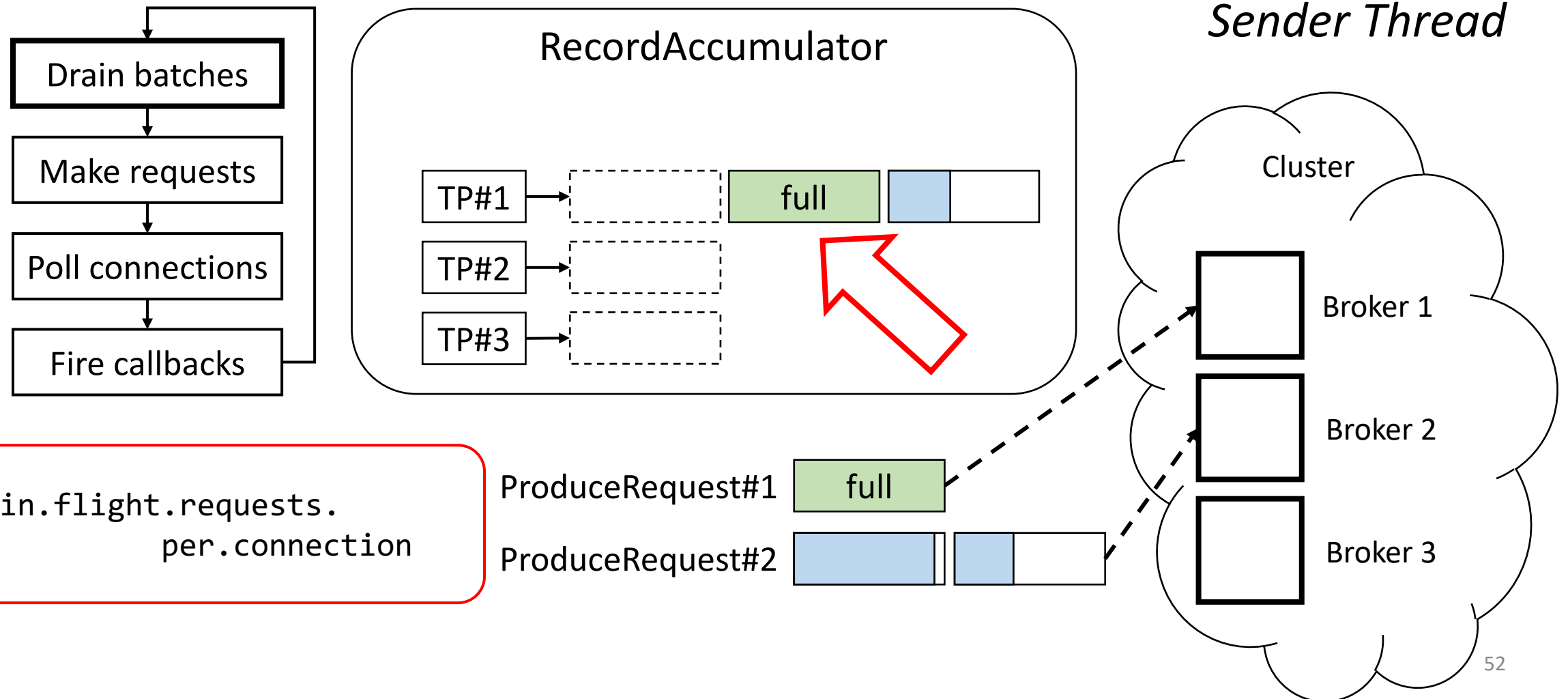
# Kafka Producer под капотом



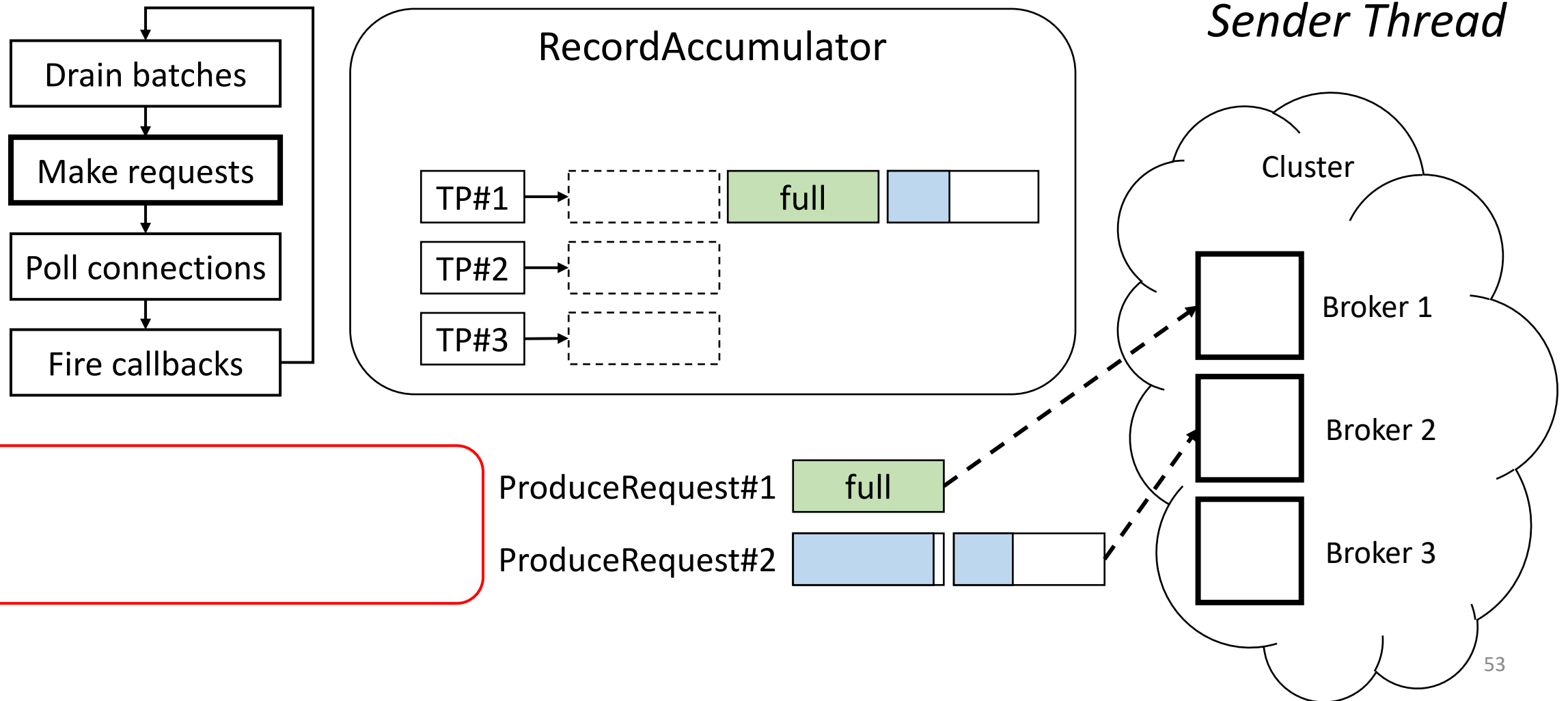
# Kafka Producer под капотом



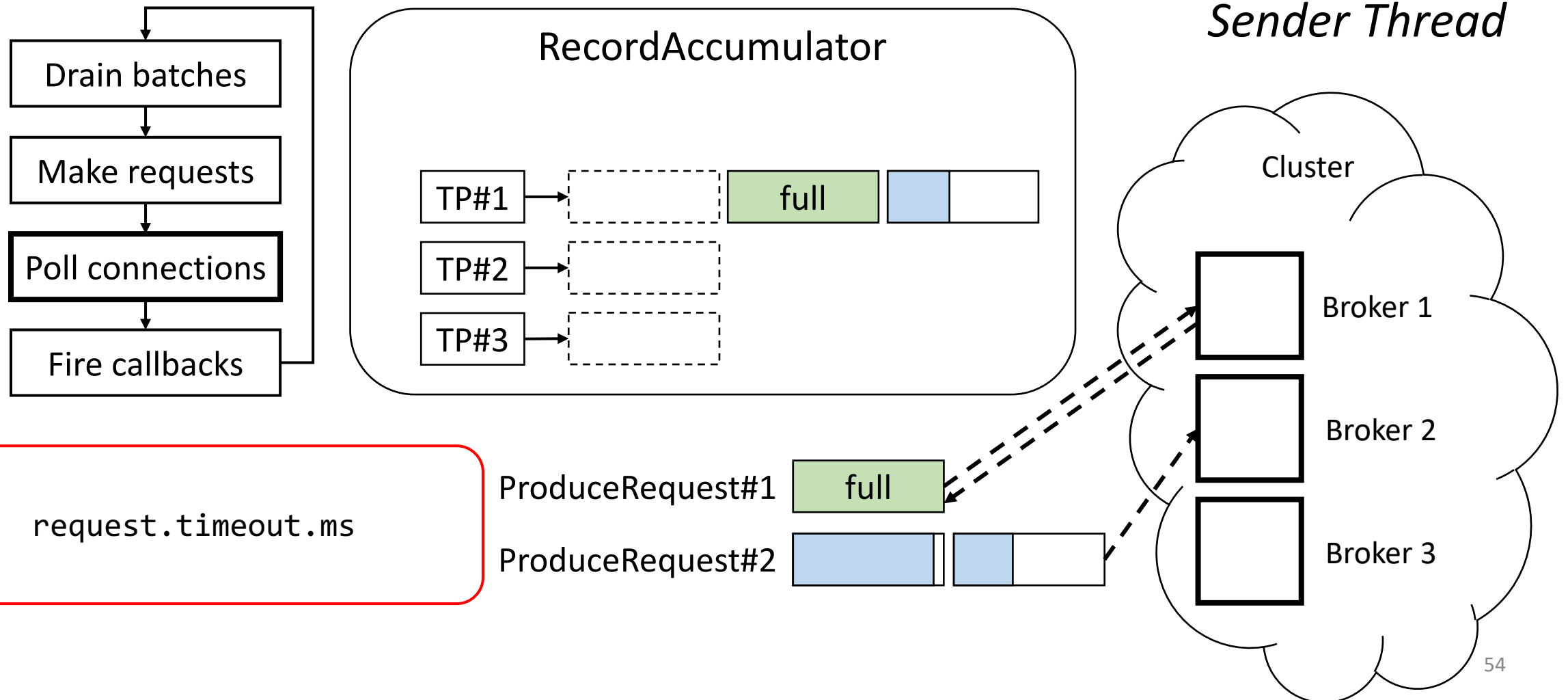
# Kafka Producer под капотом



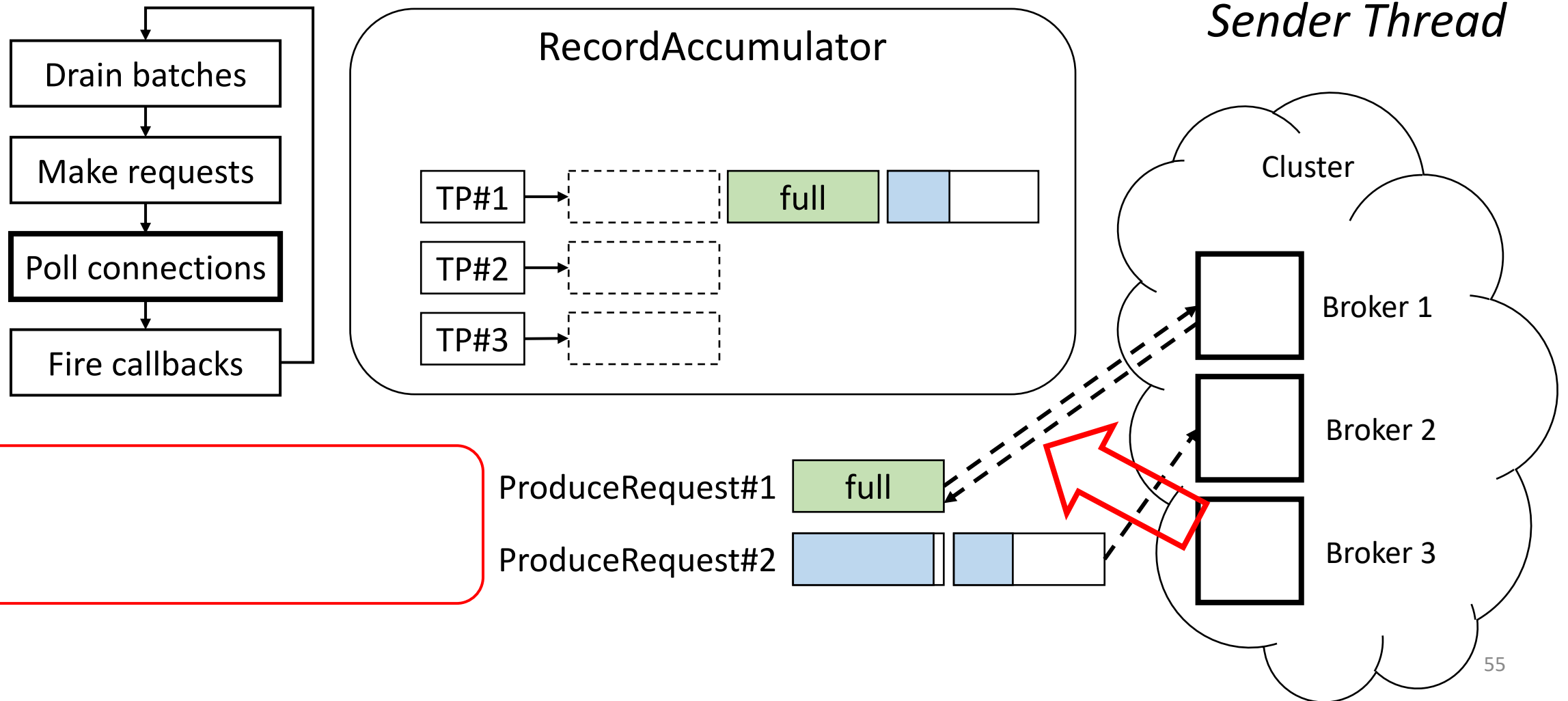
# Kafka Producer под капотом



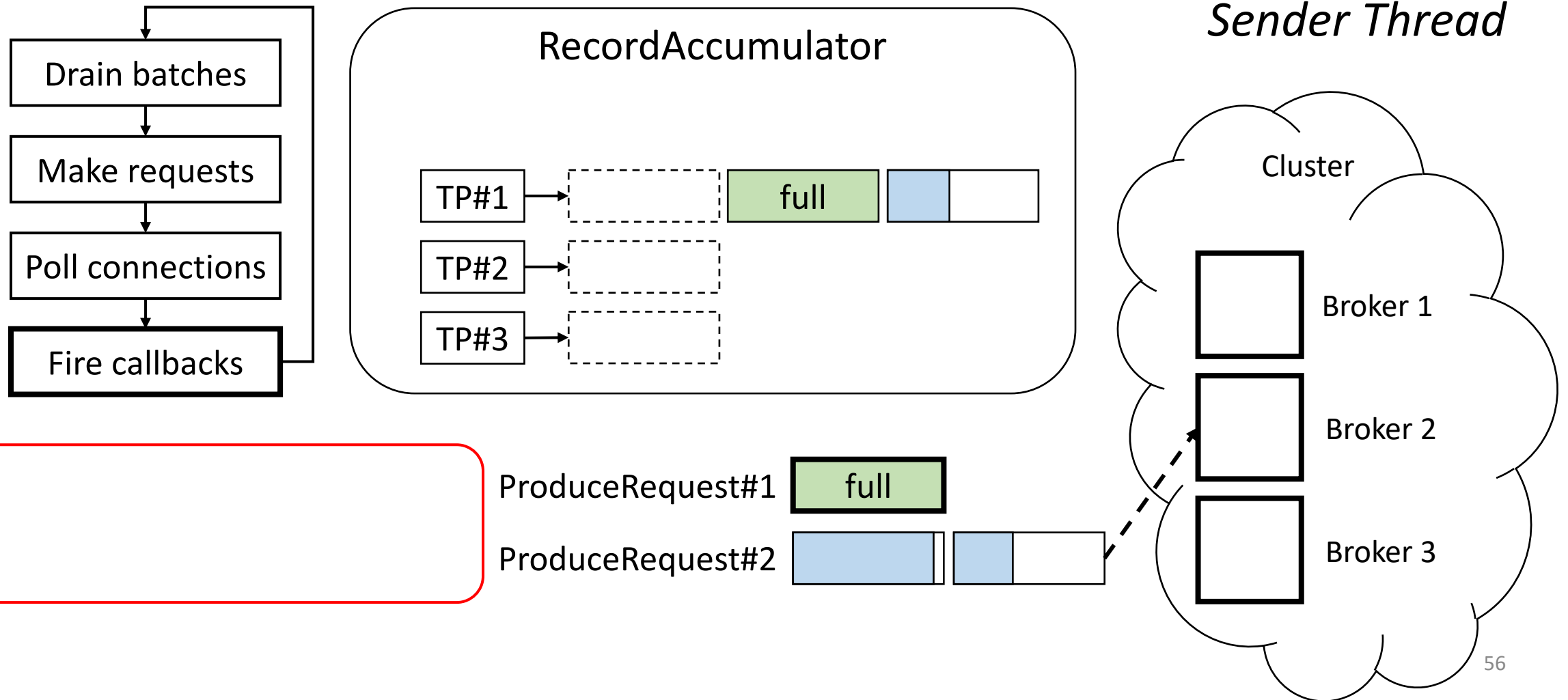
# Kafka Producer под капотом



# Kafka Producer под капотом

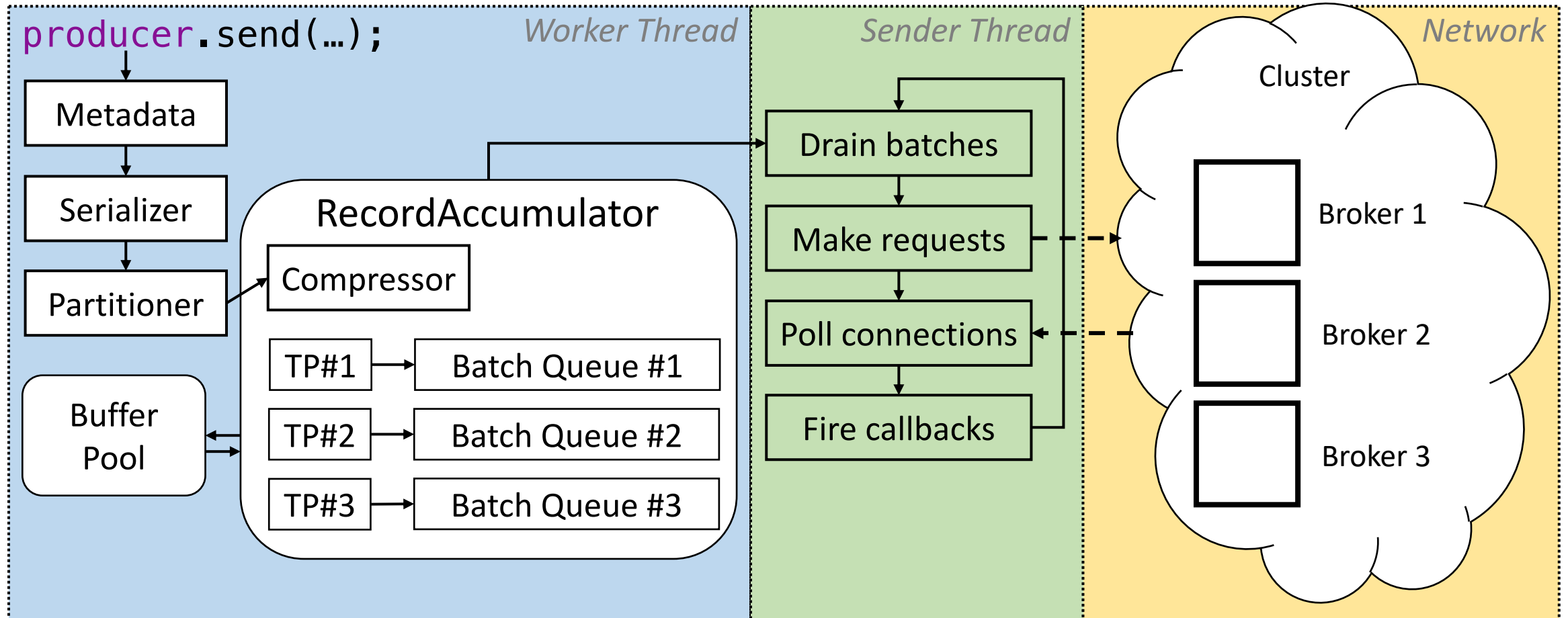


# Kafka Producer под капотом





# Принципиальная схема Kafka Producer



# Метрики производительности

- Общие метрики KafkaProducer

```
group      = producer-metrics  
client-id = producer-1
```

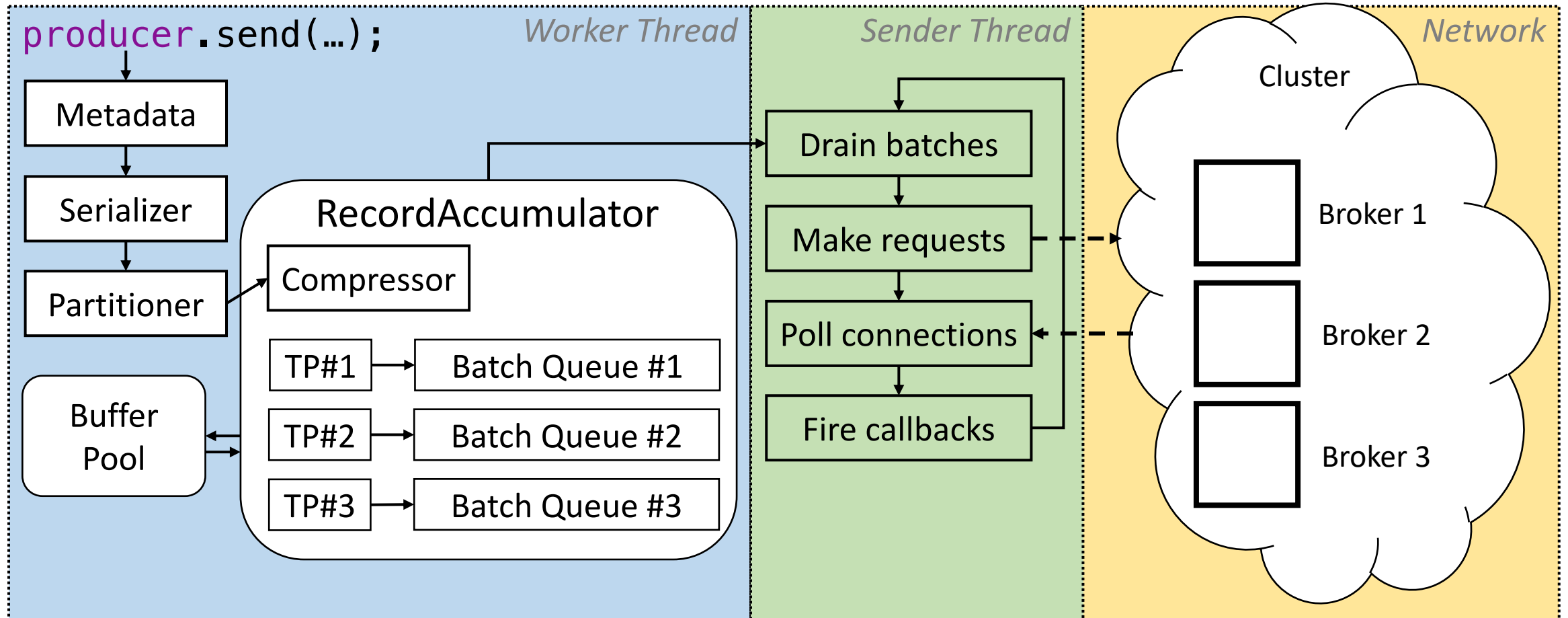
- Метрики KafkaProducer по брокерам

```
group      = producer-node-metrics  
client-id = producer-1  
node-id    = node-*
```

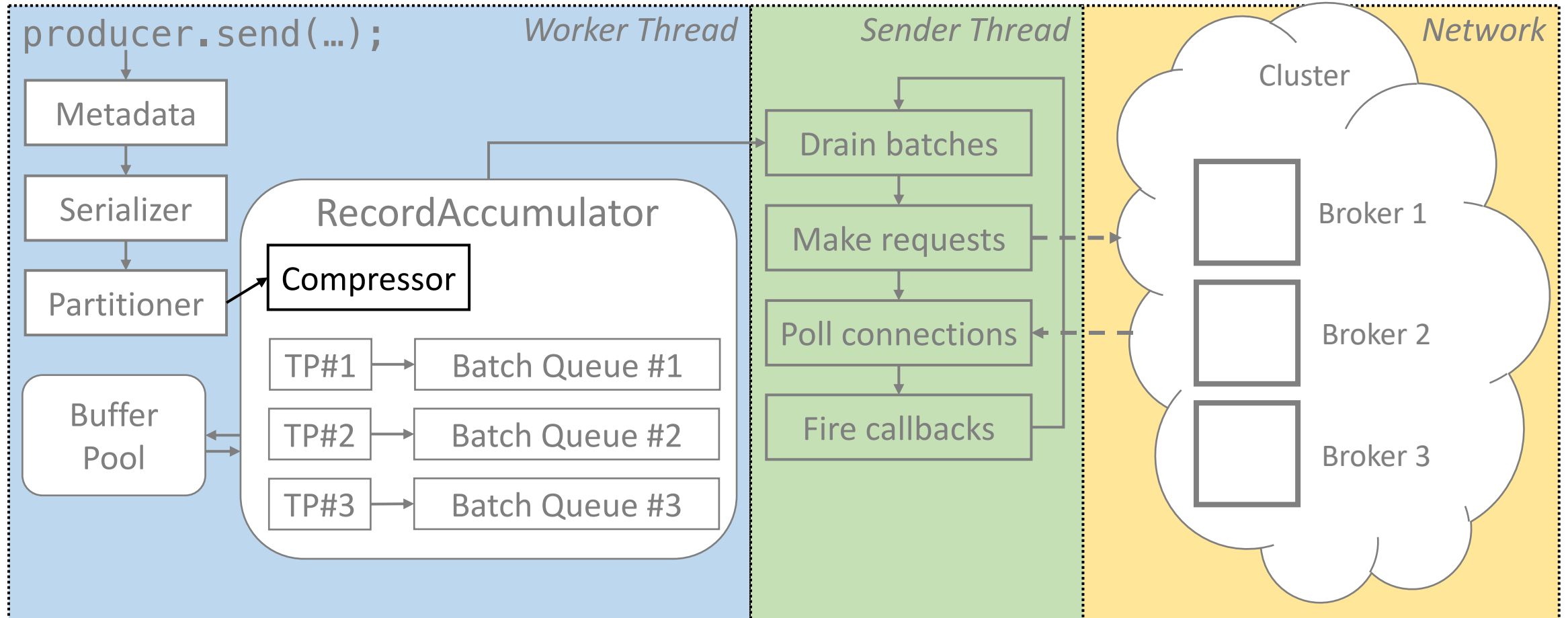
- Метрики KafkaProducer по топикам

```
group      = producer-topic-metrics  
client-id = producer-1  
topic      = *
```

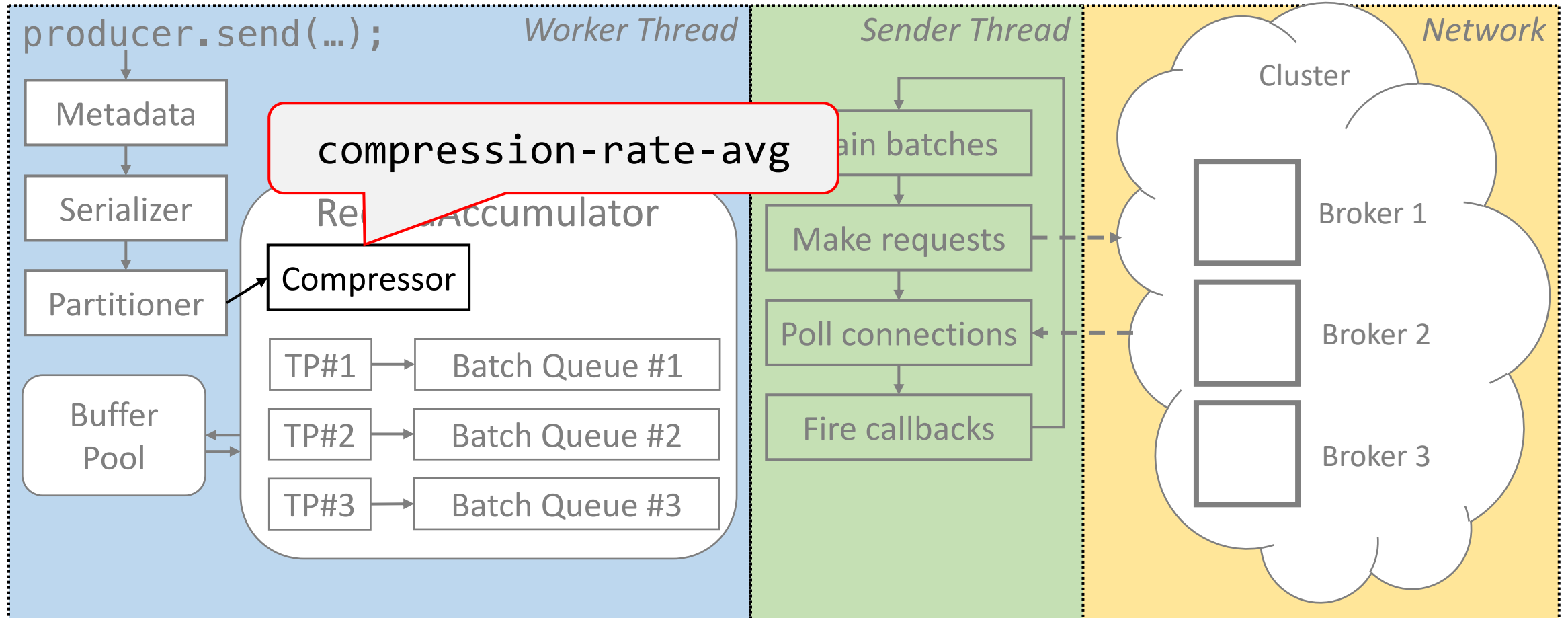
# Метрики производительности



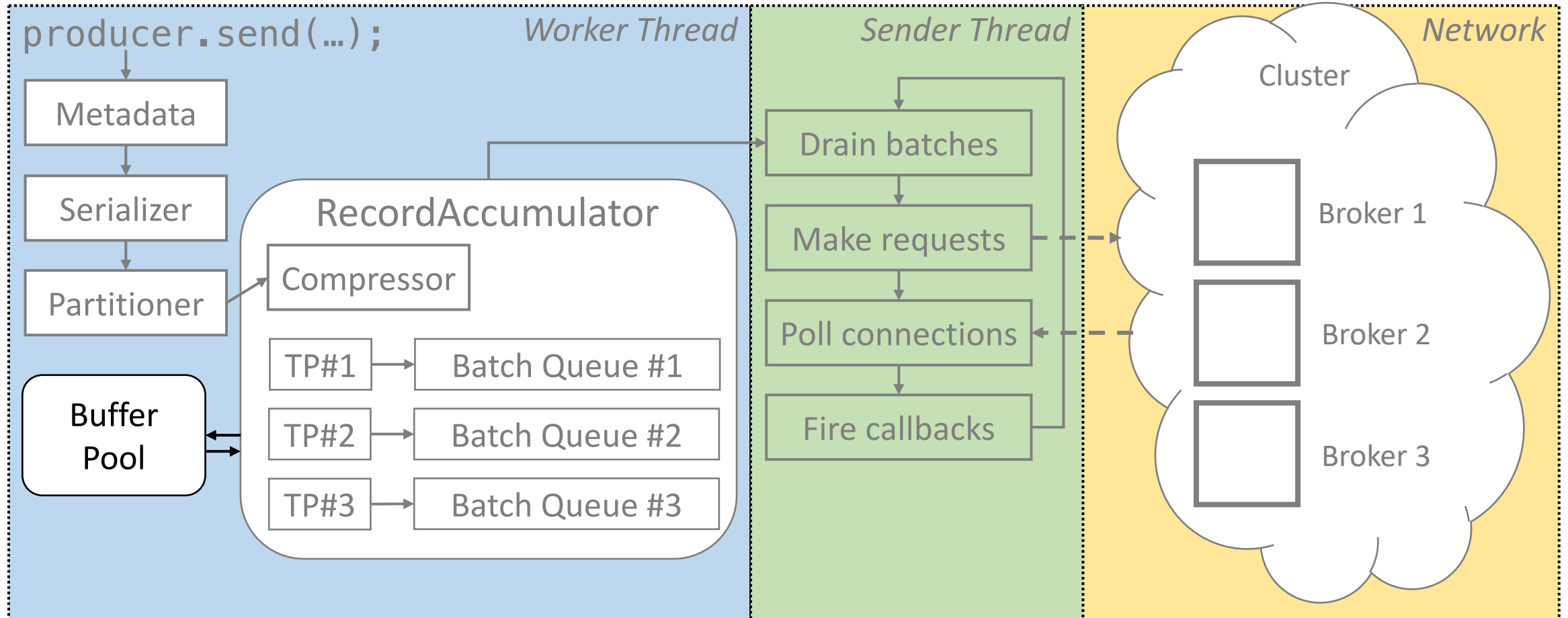
# Метрики производительности



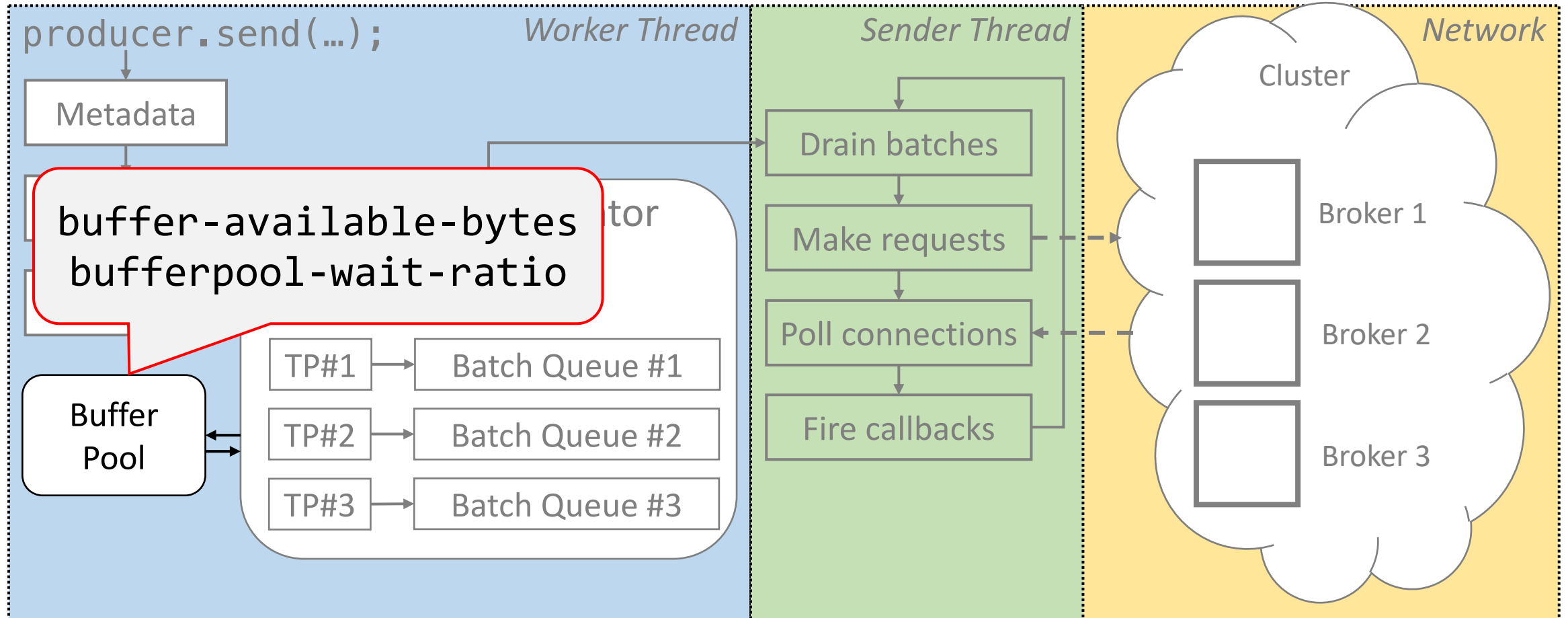
# Метрики производительности



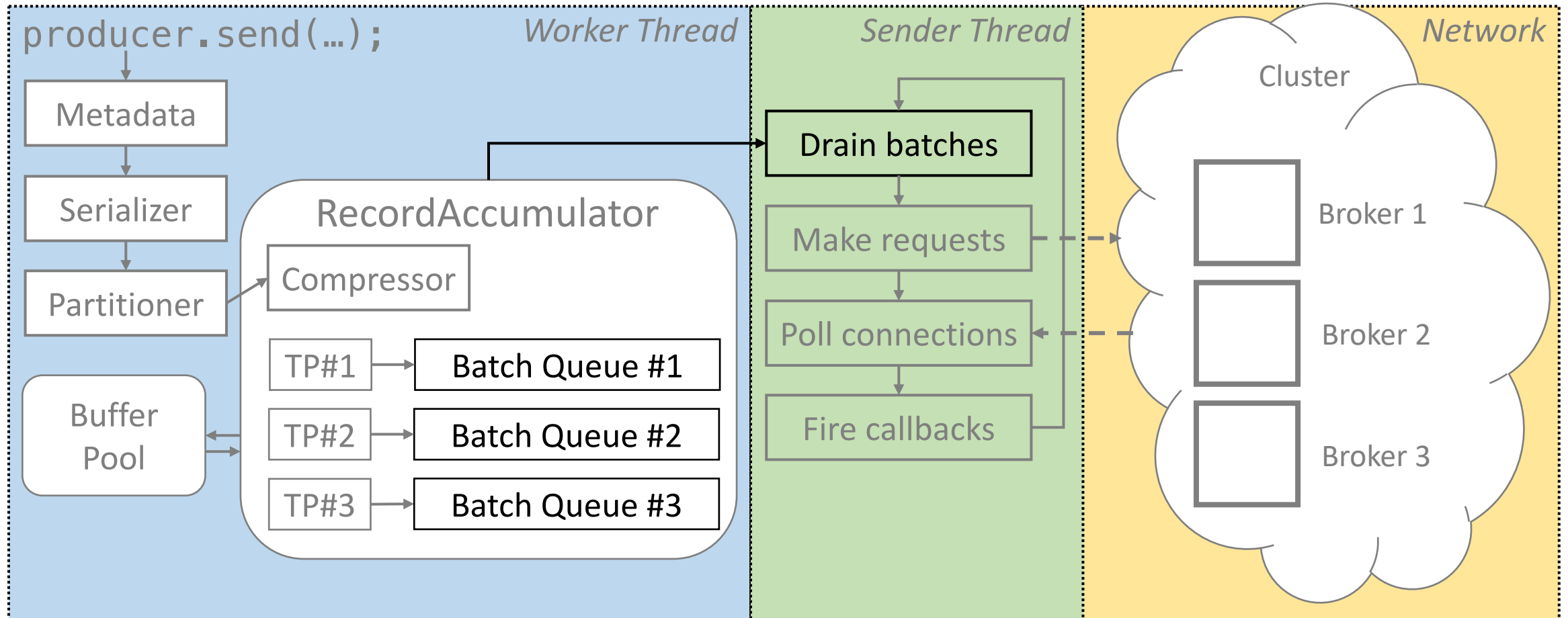
# Метрики производительности



# Метрики производительности

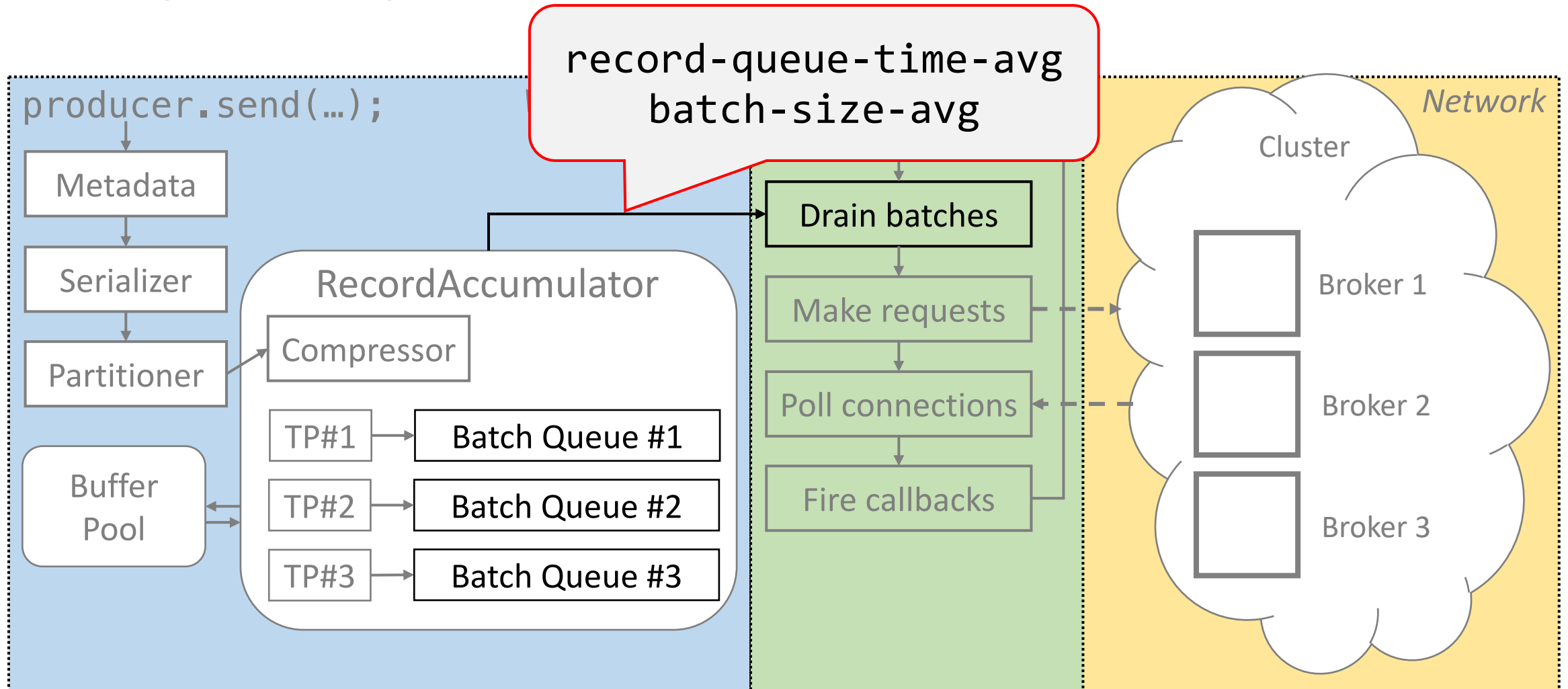


# Метрики производительности

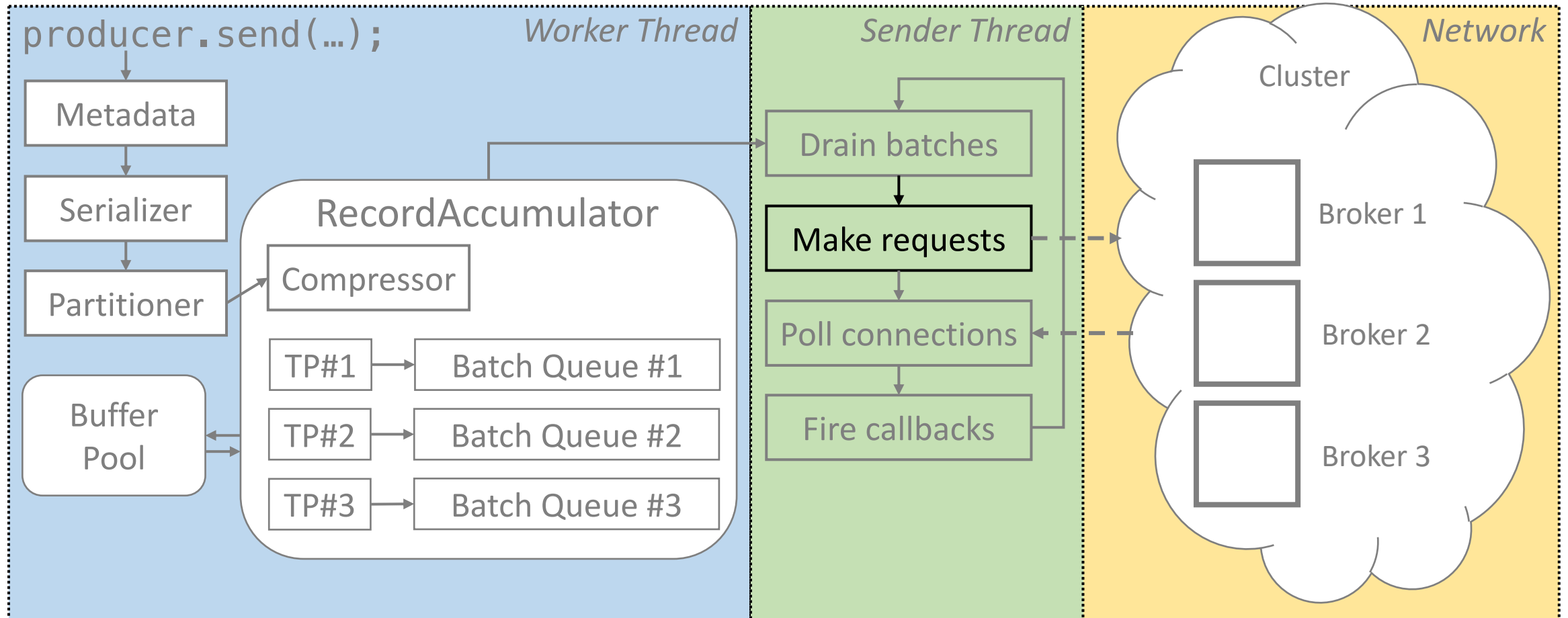




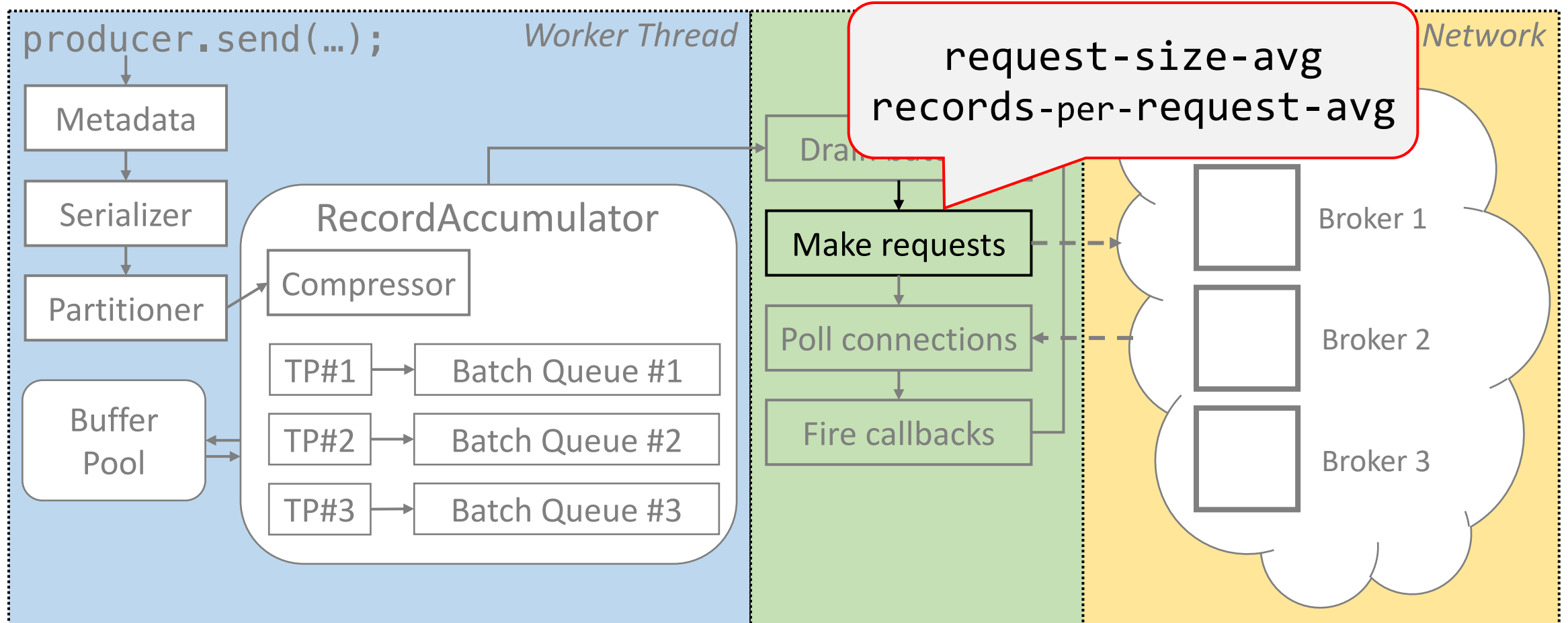
# Метрики производительности



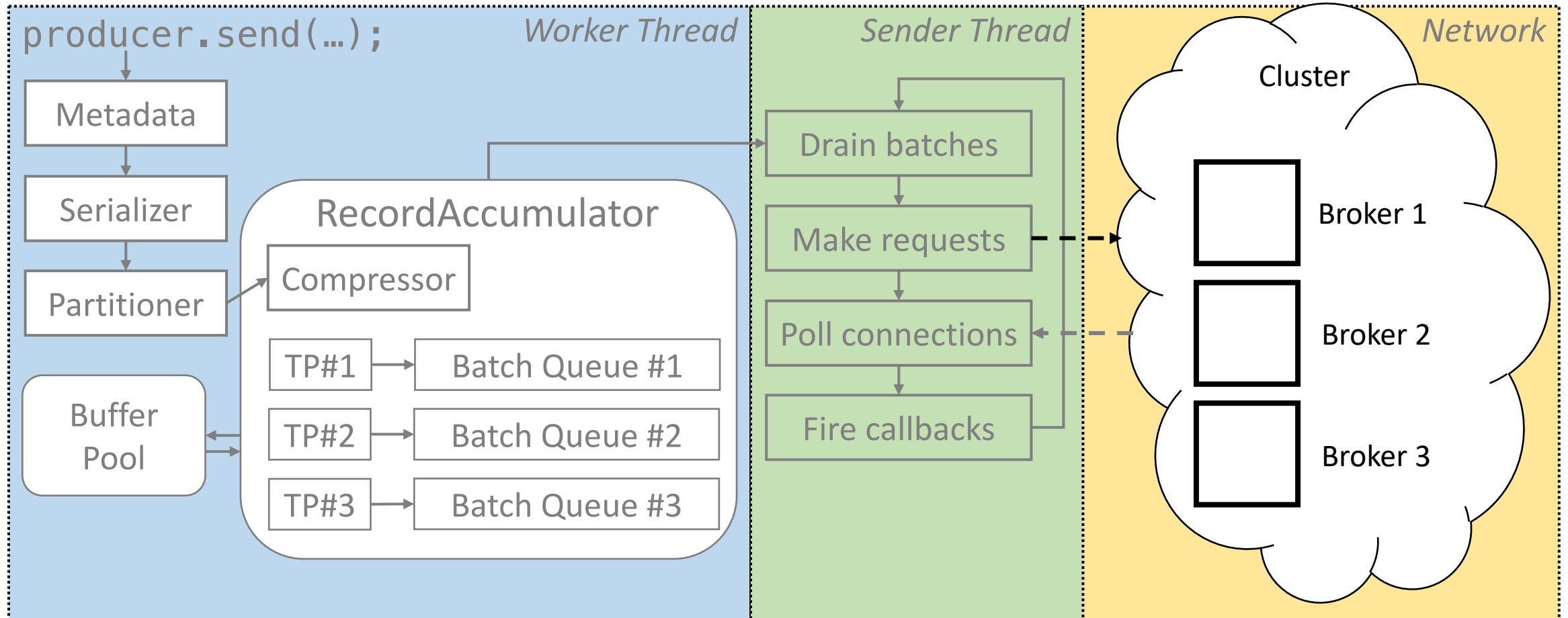
# Метрики производительности



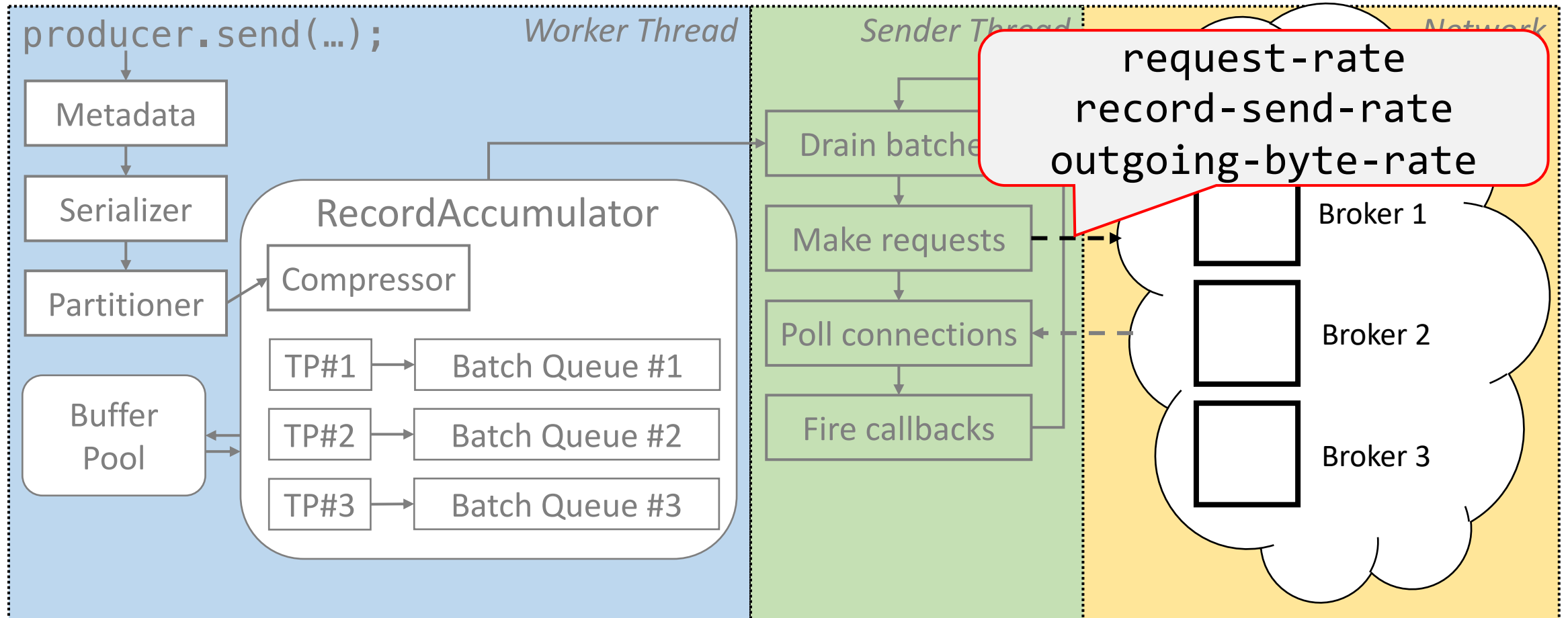
# Метрики производительности



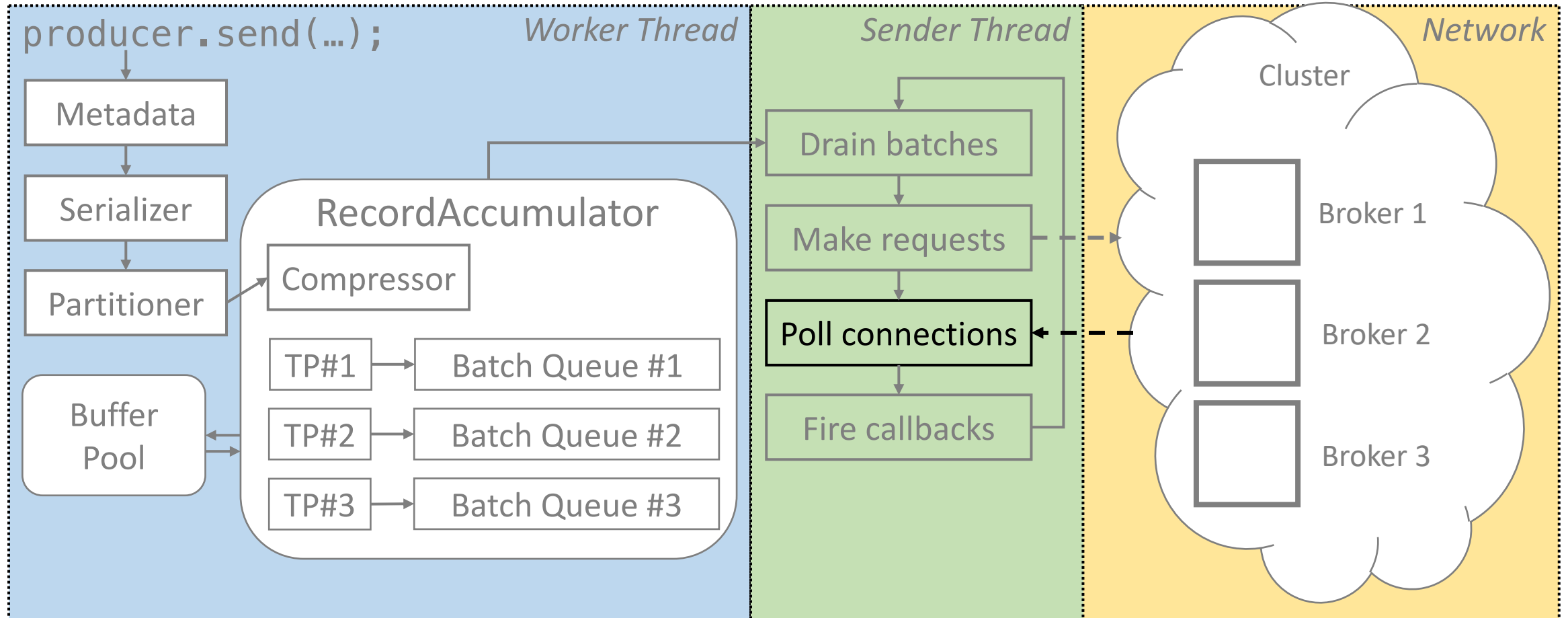
# Метрики производительности



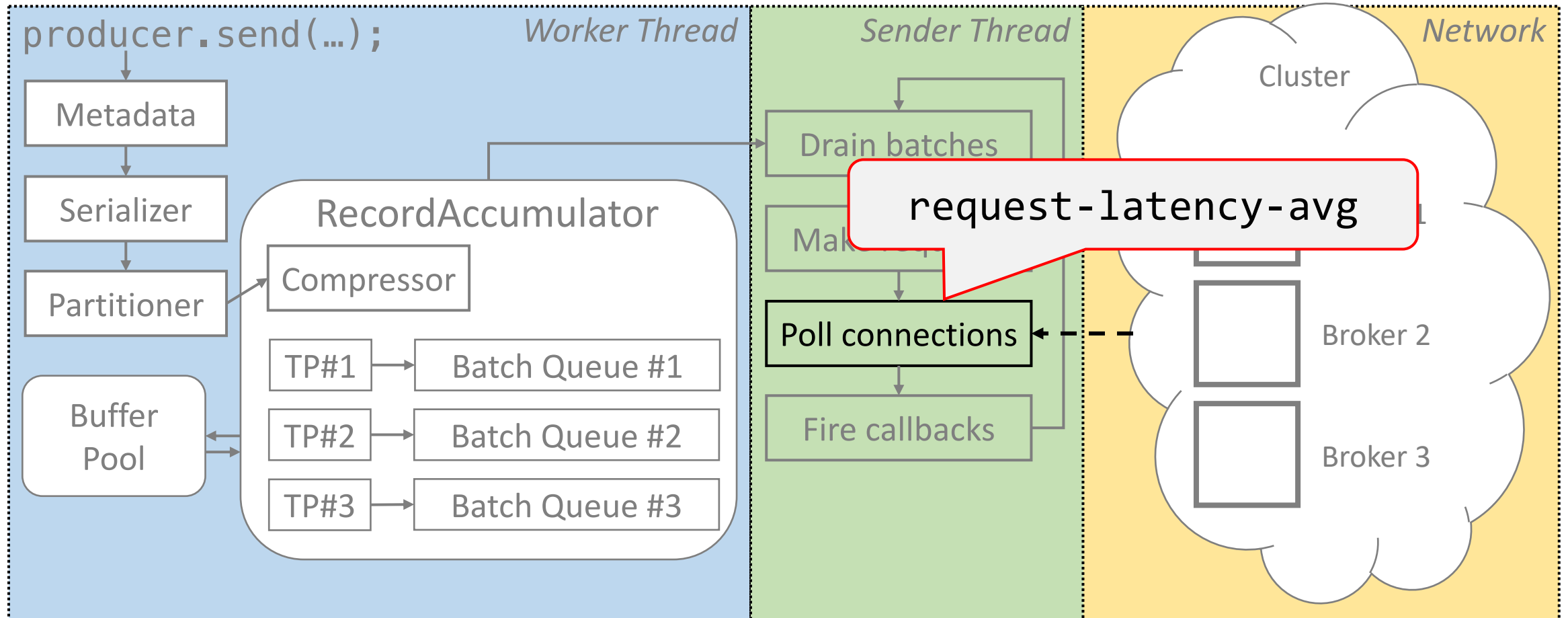
# Метрики производительности



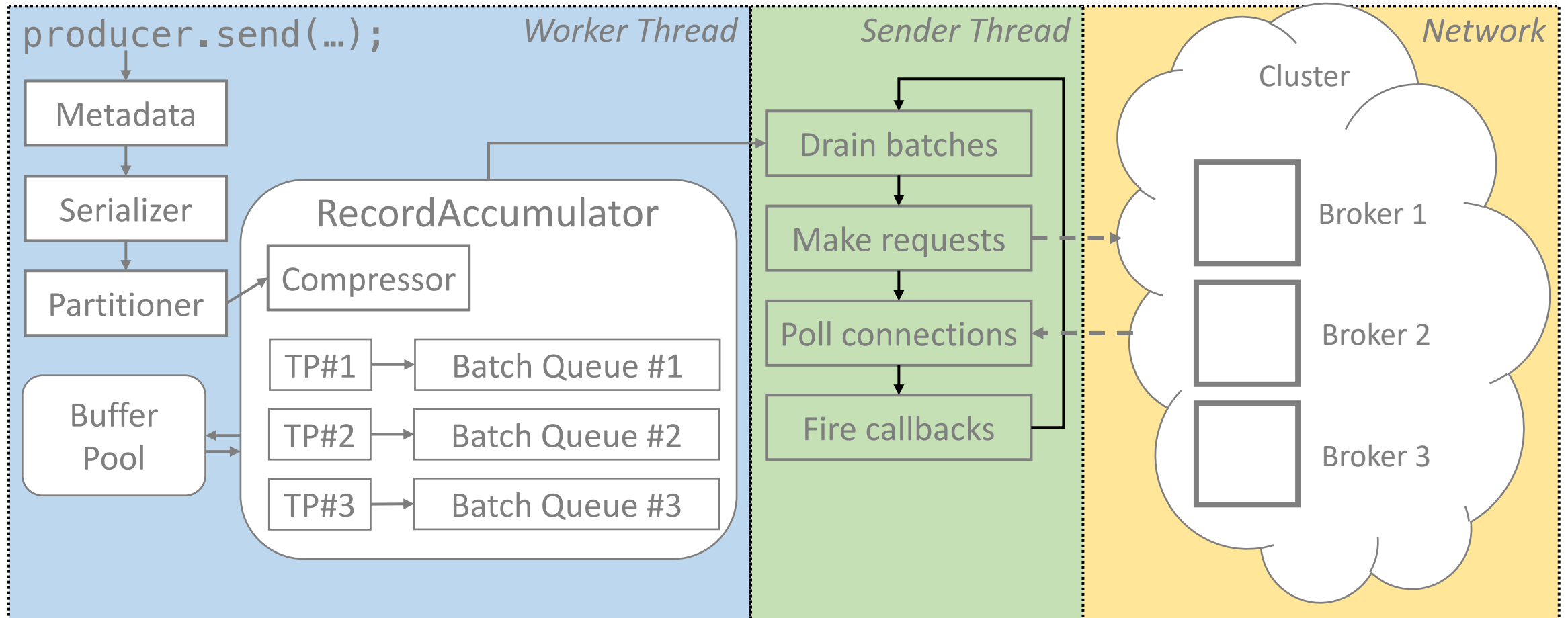
# Метрики производительности



# Метрики производительности

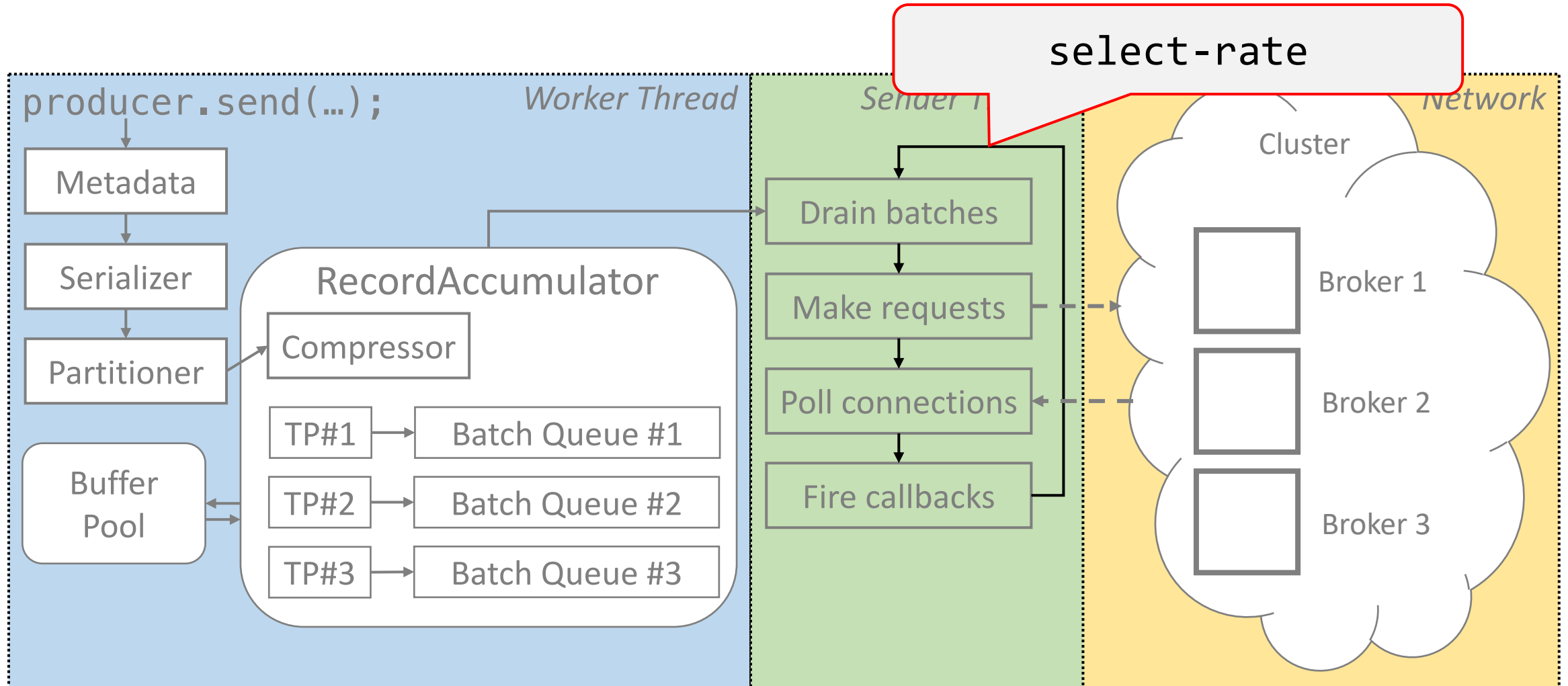


# Метрики производительности





# Метрики производительности



# Метрики производительности

`Throughput (bps) = outgoing-byte-rate / compression-rate-avg`

# Метрики производительности

**Throughput** (bps) = outgoing-byte-rate / compression-rate-avg  
= request-rate \* request-size-avg / compression-rate-avg

# Метрики производительности

**Throughput** (bps) = outgoing-byte-rate / compression-rate-avg  
= request-rate \* request-size-avg / compression-rate-avg

**Throughput** (rps) = record-send-rate

# Метрики производительности

**Throughput** (bps) = outgoing-byte-rate / compression-rate-avg  
= request-rate \* request-size-avg / compression-rate-avg

**Throughput** (rps) = record-send-rate

**Total Latency** = Worker Latency + Sender Latency + Callback Latency

# Метрики производительности

**Throughput (bps)** = `outgoing-byte-rate / compression-rate-avg`  
= `request-rate * request-size-avg / compression-rate-avg`

**Throughput (rps)** = `record-send-rate`

**Total Latency** = `Worker Latency + Sender Latency + Callback Latency`

**Worker Latency** – время выполнения `producer.send()`

# Метрики производительности

**Throughput (bps)** = `outgoing-byte-rate / compression-rate-avg`  
= `request-rate * request-size-avg / compression-rate-avg`

**Throughput (rps)** = `record-send-rate`

**Total Latency** = `Worker Latency + Sender Latency + Callback Latency`

**Worker Latency** – время выполнения `producer.send()`

**Sender Latency** – время до получения ответа от брокера

# Метрики производительности

**Throughput (bps)** = `outgoing-byte-rate / compression-rate-avg`  
= `request-rate * request-size-avg / compression-rate-avg`

**Throughput (rps)** = `record-send-rate`

**Total Latency** = `Worker Latency + Sender Latency + Callback Latency`

**Worker Latency** – время выполнения `producer.send()`

**Sender Latency** – время до получения ответа от брокера

**Callback Latency** – не влияет на End-To-End Latency



# Метрики производительности

**Sender Latency** = (record-queue-time-avg / 2) + request-latency-avg

# Поиск узких мест

## Кластер Kafka

- 3 брокера (на 3 ДЦ), версия 2.4
- RTT < 1 мс
- 128 ГБ ОЗУ, 7 HDD по 6 ТБ

# Поиск узких мест

## Кластер Kafka

- 3 брокера (на 3 ДЦ), версия 2.4
- RTT < 1 мс
- 128 ГБ ОЗУ, 7 HDD по 6 ТБ

## Producer

- 1 worker thread
- 100k RPS, 256 (почти) случайных байтов на сообщение
- 1 топик, 3 партиции

# Поиск узких мест

Эксперимент I

Настройки по умолчанию...

# Поиск узких мест

Эксперимент I

Настройки по умолчанию...

за исключением:

```
acks = all  
compression.type = lz4  
linger.ms = 5
```

# Поиск узких мест

Эксперимент I

Настройки по умолчанию...

за исключением:

```
acks = all  
compression.type = lz4  
linger.ms = 5
```

# Поиск узких мест

Эксперимент I

Настройки по умолчанию...

за исключением:

```
acks = all  
compression.type = lz4  
linger.ms = 5
```

# Поиск узких мест

Эксперимент I

Настройки по умолчанию...

за исключением:

```
acks = all  
compression.type = lz4  
linger.ms = 5
```



# Поиск узких мест

## Эксперимент I

Throughput, rps	Worker Latency, $\mu$ s	Sender Latency, ms

# Поиск узких мест

## Эксперимент I

Throughput, rps	Worker Latency, $\mu$ s	Sender Latency, ms
83 550 $\pm$ 1 590		

# Поиск узких мест

## Эксперимент I

Throughput, rps	Worker Latency, $\mu$ s	Sender Latency, ms
83 550 $\pm$ 1 590		

НО ПРОСИЛИ 100 000! ☹️

# Поиск узких мест

## Эксперимент I

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
83 550 $\pm$ 1 590	11.6 $\pm$ 3.5 $\mu s$	

# Поиск узких мест

## Эксперимент I

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
83 550 $\pm$ 1 590	11.6 $\pm$ 3.5 $\mu s$	

$$1 / 83550 \approx 0.000012$$

# Поиск узких мест

## Эксперимент I

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
83 550 $\pm$ 1 590	11.6 $\pm$ 3.5 $\mu s$	

$$\begin{aligned} 1 / 83550 &\approx 0.000012 \\ &= 12 \mu s \end{aligned}$$

# Поиск узких мест

## Эксперимент I

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
83 550 $\pm$ 1 590	11.6 $\pm$ 3.5 $\mu s$	7 200 $\pm$ 100 ms

# Поиск узких мест

## Эксперимент I

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
83 550 $\pm$ 1 590	11.6 $\pm$ 3.5 $\mu s$	7 200 $\pm$ 100 ms

СЕМЬ СЕКУНД!!!



# Поиск узких мест

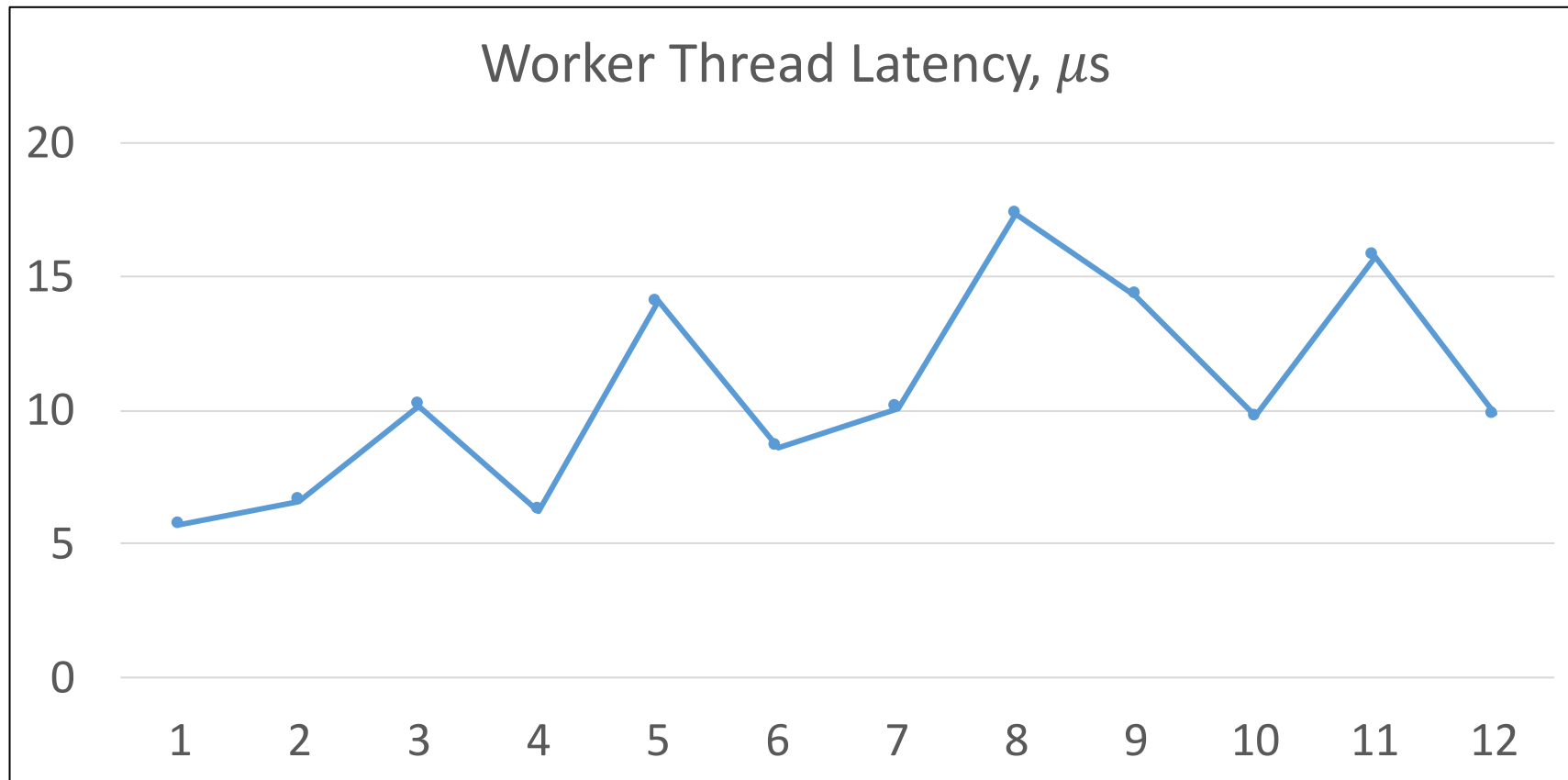
## Эксперимент I

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
83 550 $\pm$ 1 590	11.6 $\pm$ 3.5 $\mu s$	7 200 $\pm$ 100 ms

**СЕМЬ!!!  
СЕКУНД!!!**

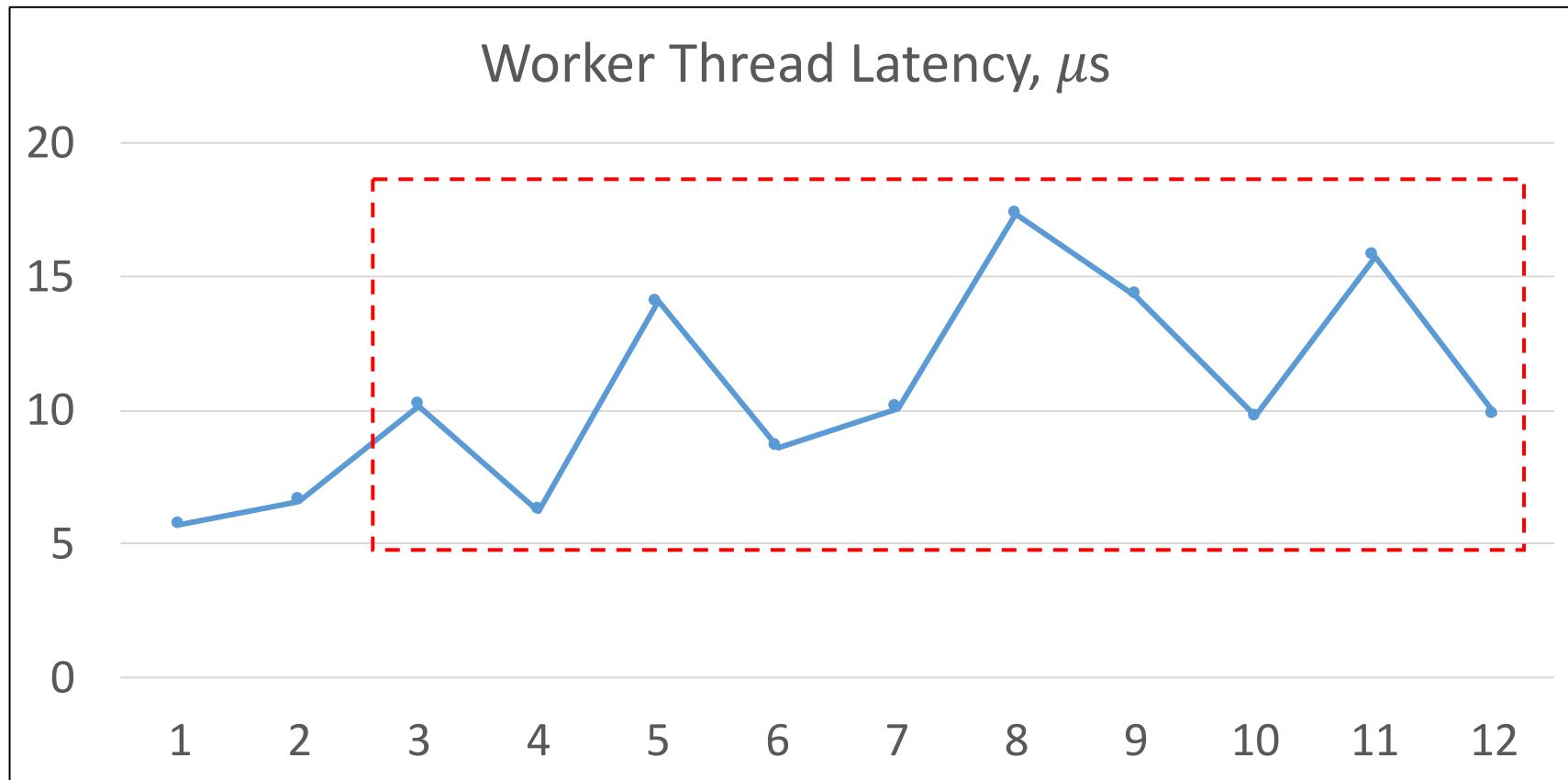
# Поиск узких мест

## Эксперимент I



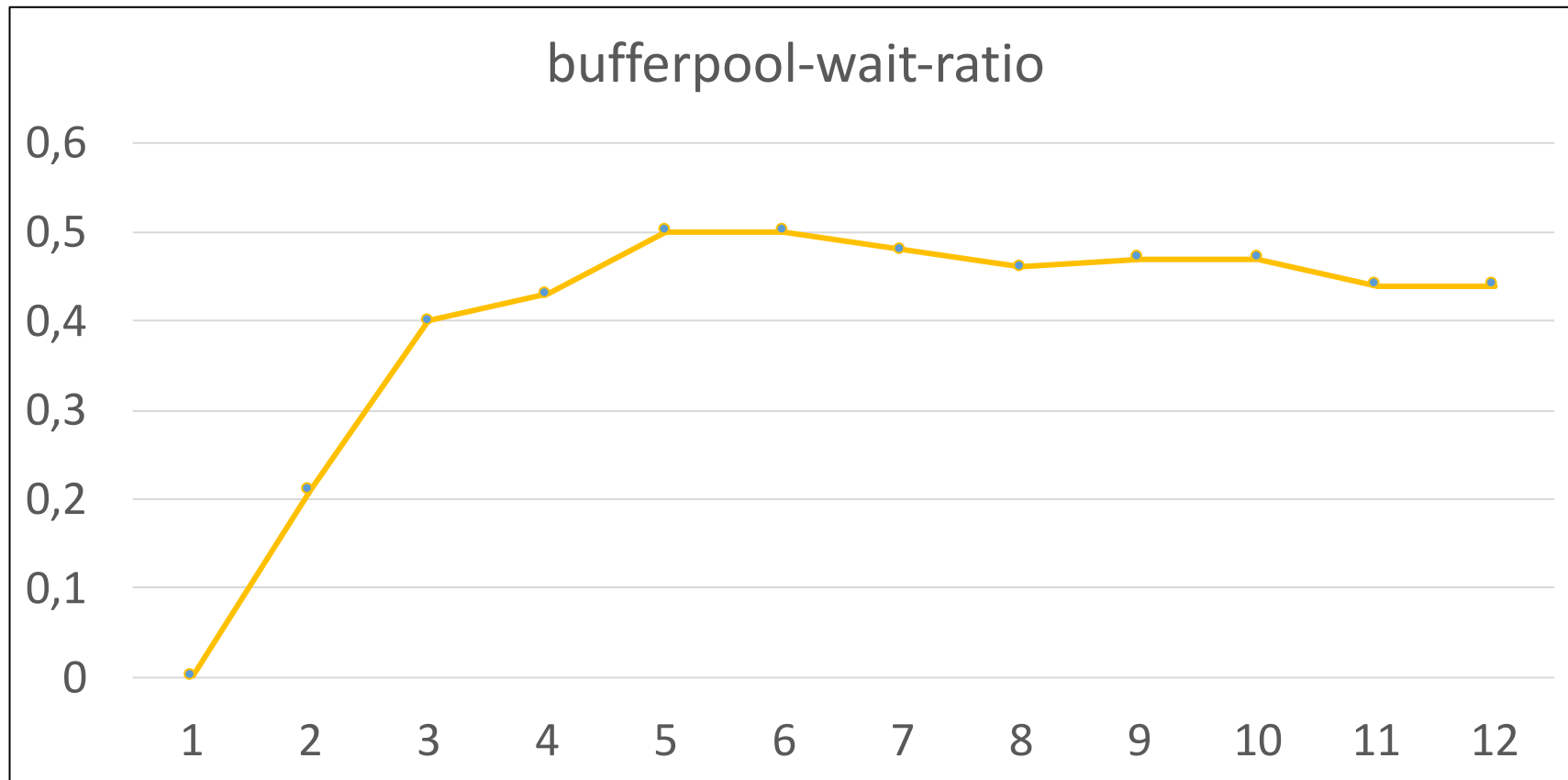
# Поиск узких мест

## Эксперимент I



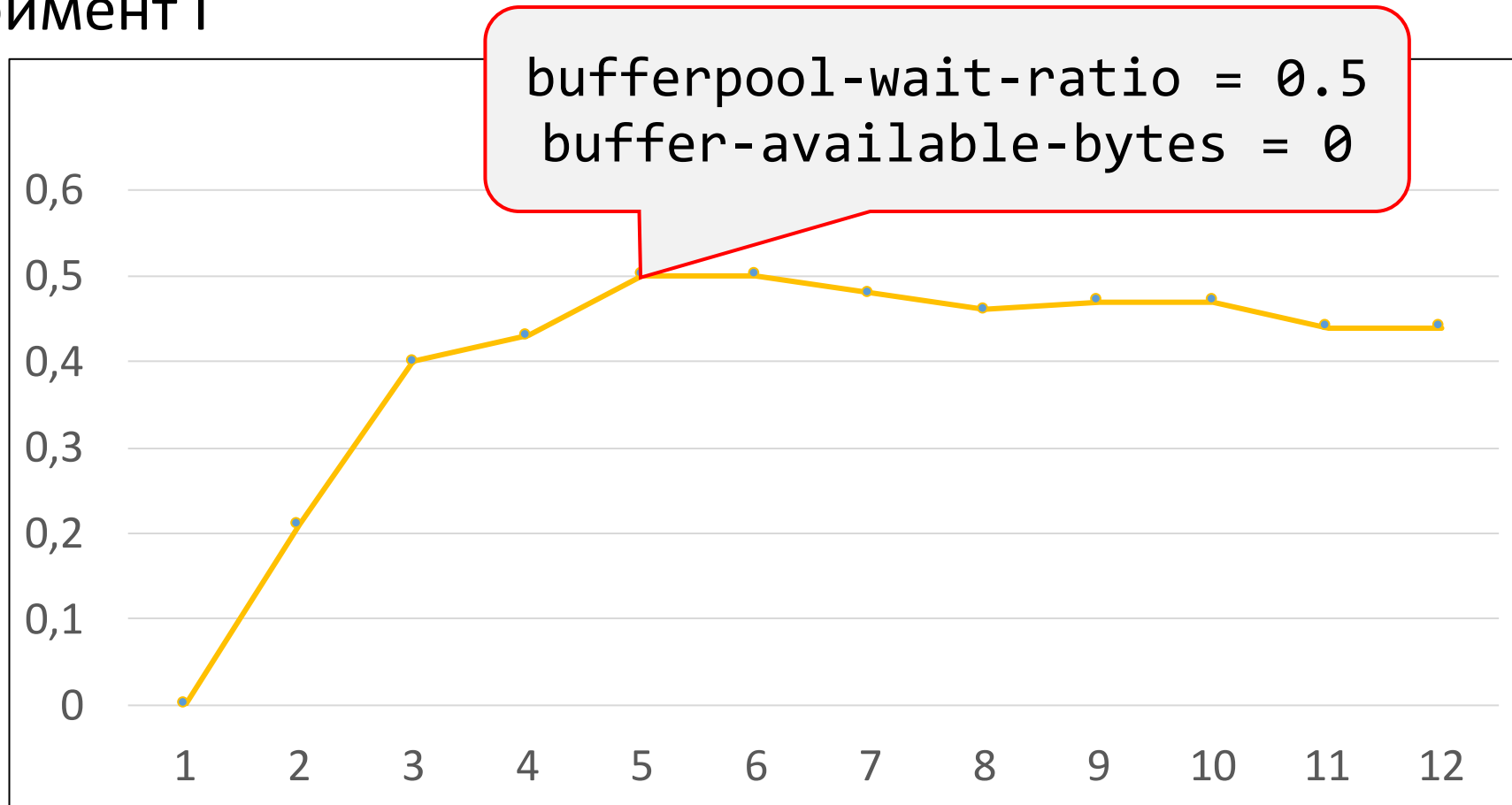
# Поиск узких мест

## Эксперимент I

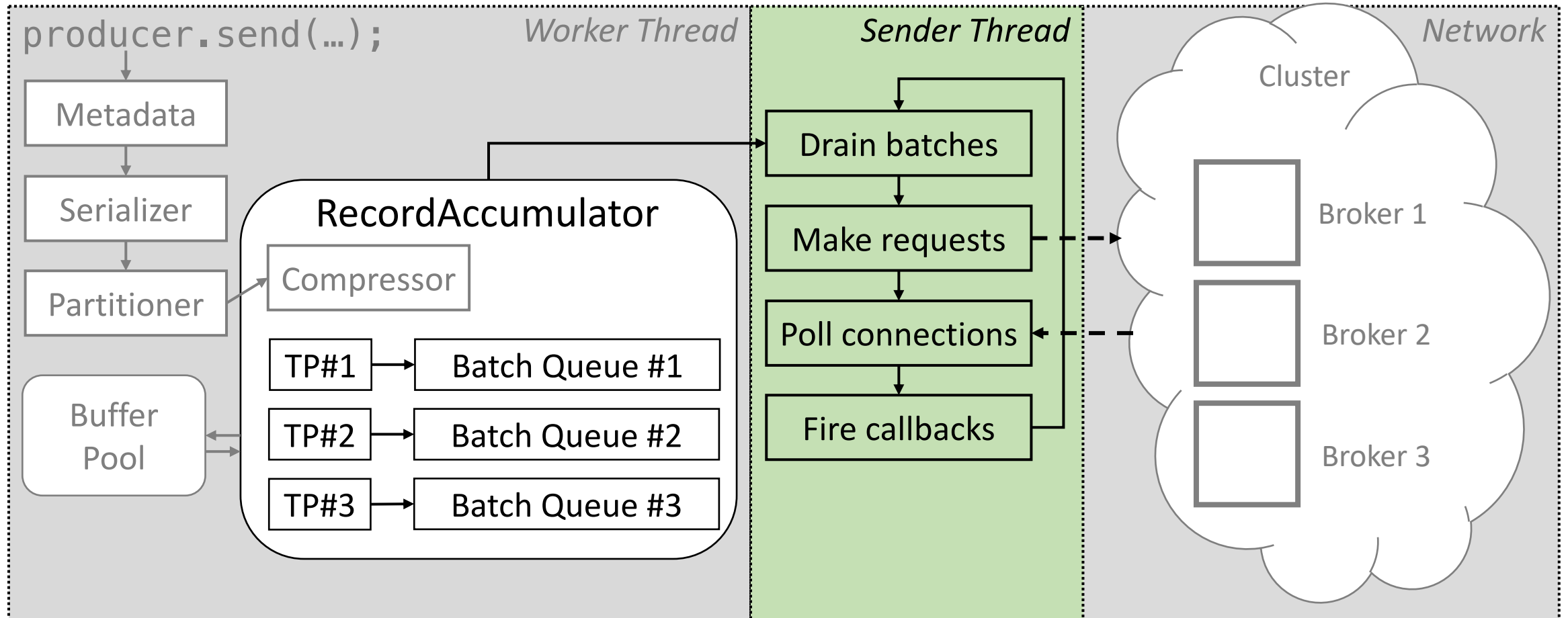


# Поиск узких мест

## Эксперимент I



# Узкие места – Sender Thread



# Поиск узких мест

Почему упёрлись в Sender Thread?

# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```



# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```

**Max Request Rate**

# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```

**Max Request Rate = 1000 / request-latency-avg**

# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```

```
Max Request Rate = 1000 / request-latency-avg  
* max.in.flight.requests.per.connection
```

# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```

```
Max Request Rate = 1000 / request-latency-avg  
* max.in.flight.requests.per.connection * количество брокеров
```

# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```

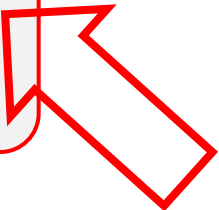
```
Max Request Rate = 1000 / request-latency-avg  
* max.in.flight.requests.per.connection * количество брокеров
```

```
Max Request Rate =
```

# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```



**Max Request Rate** =  $1000 / \text{request-latency-avg}$   
\* `max.in.flight.requests.per.connection` \* количество брокеров

**Max Request Rate** =  $1000 / 9.8$

# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```

**Max Request Rate = 1000 / request-latency-avg**  
**\* max.in.flight.requests.per.connection \* количество брокеров**



```
Max Request Rate = 1000 / 9.8  
* 5
```

# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```

**Max Request Rate** =  $1000 / \text{request-latency-avg}$   
\* `max.in.flight.requests.per.connection` \* количество брокеров

**Max Request Rate** =  $1000 / 9.8$   
\* 5 \* 3

3 брокера в кластере



# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```

**Max Request Rate = 1000 / request-latency-avg**  
**\* max.in.flight.requests.per.connection \* количество брокеров**

**Max Request Rate = 1000 / 9.8**  
**\* 5 \* 3 ≈ 1531**

Теоретический максимум!

# Поиск узких мест

Почему упёрлись в Sender Thread?

```
request-latency-avg = 9.8 ms  
request-rate = 1391  
batch-size-avg = 15066 bytes
```

**Max Request Rate = 1000 / request-latency-avg**  
**\* max.in.flight.requests.per.connection \* количество брокеров**

**Max Request Rate = 1000 / 9.8**  
**\* 5 \* 3 ≈ 1531**

Теоретический максимум!

# Поиск узких мест

Что делать?

# Поиск узких мест

Что делать?

Throughput (bps) =  
request-rate \* request-size-avg / compression-rate-avg

# Поиск узких мест

Что делать?

Throughput (bps) =  
**request-rate** \* request-size-avg / compression-rate-avg

# Поиск узких мест

Что делать?

Throughput (bps) =  
request-rate \* **request-size-avg** / compression-rate-avg

# Поиск узких мест

Что делать?

Throughput (bps) =  
`request-rate * request-size-avg / compression-rate-avg`

# Поиск узких мест

Что делать?

Throughput (bps) =  
request-rate \* **request-size-avg** / compression-rate-avg



# Поиск узких мест

Что делать?

Throughput (bps) =

`request-rate * request-size-avg / compression-rate-avg`

- увеличить `linger.ms`

# Поиск узких мест

Что делать?

Throughput (bps) =

`request-rate * request-size-avg / compression-rate-avg`

- увеличить `linger.ms`
- увеличить `batch.size`

# Поиск узких мест

Что делать?

Throughput (bps) =

`request-rate * request-size-avg / compression-rate-avg`

- увеличить `linger.ms`
- увеличить `batch.size`
- добавить партиций в топик

# Поиск узких мест

Что делать?

Throughput (bps) =

`request-rate * request-size-avg / compression-rate-avg`

- **увеличить `linger.ms`**
- **увеличить `batch.size`**
- **добавить партиций в топик**

НЕ ПОМОЖЕТ!

- пачки заполнены (`batch-size-avg`)
- `record-queue-time-avg` больше **14** секунд

# Поиск узких мест

Что делать?

Throughput (bps) =

`request-rate * request-size-avg / compression-rate-avg`

- увеличить `linger.ms`
- **увеличить `batch.size`**
- добавить партиций в топик

# Поиск узких мест

Эксперимент II и III

batch.size, kbytes	Worker Latency, $\mu s$	Sender Latency, ms
16	$11.6 \pm 3.5 \mu s$	$7\,200 \pm 100 \text{ ms}$

# Поиск узких мест

Эксперимент II и III

batch.size, kbytes	Worker Latency, $\mu\text{s}$	Sender Latency, ms
16	$11.6 \pm 3.5 \mu\text{s}$	$7\,200 \pm 100 \text{ ms}$
<b>32</b>	<b><math>3.5 \pm 0.3 \mu\text{s}</math></b> ↓↓	<b><math>5.1 \pm 0.7 \text{ ms}</math></b> ↓↓
<b>64</b>	<b><math>3.6 \pm 0.7 \mu\text{s}</math></b> ↓↓	<b><math>5.5 \pm 0.6 \text{ ms}</math></b> ↓↓

И ПОЛУЧИЛИ СВОИ 100 000! 😊

# Поиск узких мест

## Эксперимент IV

- Ослабим ограничение на RPS до 500 000
- Оставим `batch.size = 64 kbytes` с предыдущего теста



# Поиск узких мест

## Эксперимент IV

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms

# Поиск узких мест

## Эксперимент IV

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms

$$1\ 000\ 000 / 3.6 \approx 277\ 777$$

# Поиск узких мест

## Эксперимент IV

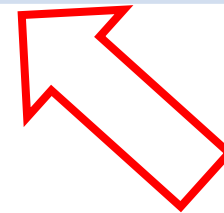
Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms

$$1\ 000\ 000 / 3.6 \approx 277\ 777$$

# Поиск узких мест

## Эксперимент IV

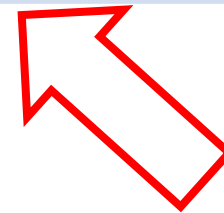
Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms



# Поиск узких мест

## Эксперимент IV

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms



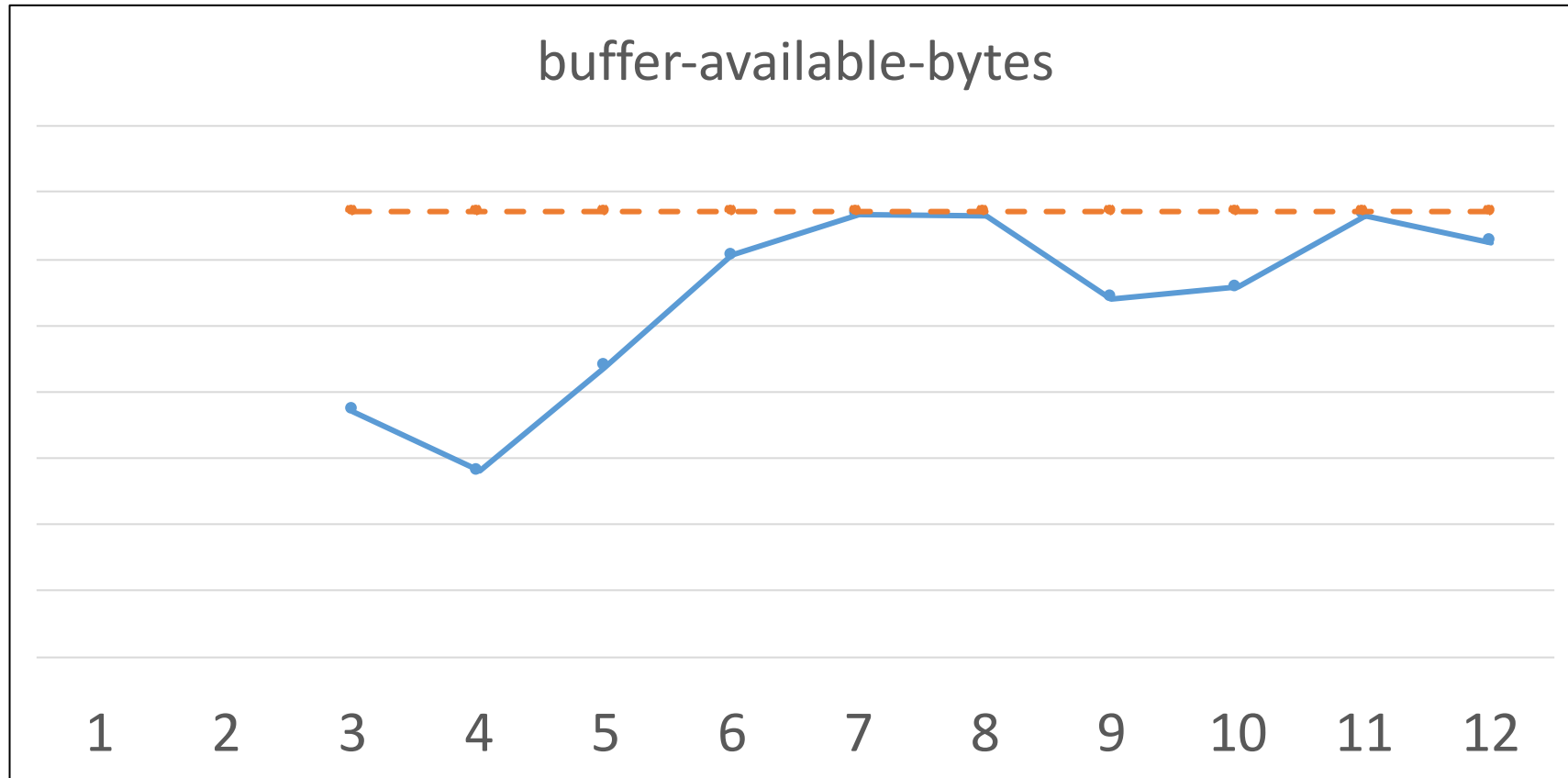
# Поиск узких мест

## Эксперимент IV

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	45 <del>ms</del>

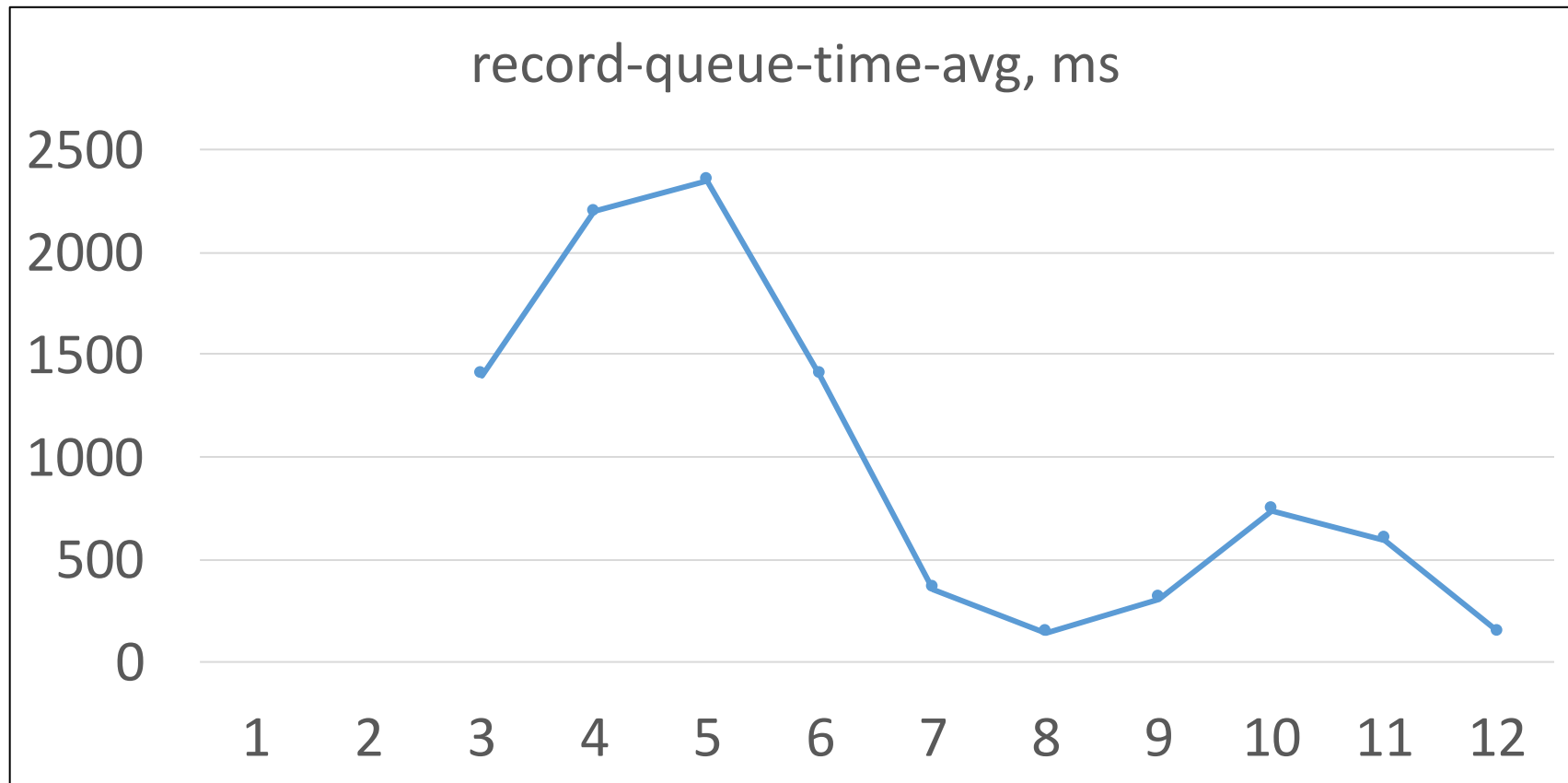
# Поиск узких мест

## Эксперимент IV



# Поиск узких мест

## Эксперимент IV





# Поиск узких мест

## Эксперимент IV

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms

```
record-queue-time-avg = 963 ms  
request-latency-avg = 11 ms  
request-rate = 1159
```

# Поиск узких мест

## Эксперимент IV

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms

```
record-queue-time-avg = 963 ms  
request-latency-avg = 11 ms  
request-rate = 1159
```

$$1000 / 11 * 5 * 3 \approx 1364$$

# Поиск узких мест

## Эксперимент IV

Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms

record-queue-time-avg = 963 ms  
request-latency-avg = 11 ms  
request-rate = **1159**

$$1000 / 11 * 5 * 3 \approx \mathbf{1364}$$

# Поиск узких мест

Эксперимент V

- Увеличим количество партиций в топике с **3** до **6**

# Поиск узких мест

## Эксперимент V

Partitions	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
3	282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms

```
record-queue-time-avg = 963 ms  
request-latency-avg = 11 ms  
request-rate = 1159
```

# Поиск узких мест

## Эксперимент V

Partitions	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
3	282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms
6	343 500 $\pm$ 6 100 $\uparrow$	2.4 $\pm$ 0.2 $\mu s$ $\downarrow$	29.0 $\pm$ 4.7 ms $\downarrow\downarrow$

record-queue-time-avg = 963 ms  
request-latency-avg = 11 ms  
request-rate = 1159

$\Rightarrow$

record-queue-time-avg = 34.5 ms  
request-latency-avg = 11.8 ms  
request-rate = 1029

# Поиск узких мест

## Эксперимент V

Partitions	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
3	282 600 $\pm$ 7 200	3.6 $\pm$ 0.9 $\mu s$	492 $\pm$ 413 ms
6	343 500 $\pm$ 6 100 $\uparrow$	2.4 $\pm$ 0.2 $\mu s$ $\downarrow$	29.0 $\pm$ 4.7 ms $\downarrow\downarrow$

record-queue-time-avg = 963 ms  
request-latency-avg = 11 ms  
request-rate = 1159

$\Rightarrow$

record-queue-time-avg = **34.5 ms**  
request-latency-avg = 11.8 ms  
request-rate = 1029

# Поиск узких мест

Упёрлись в Worker Thread



# Поиск узких мест

Упёрлись в Worker Thread

Попробуем увеличить количество потоков

# Поиск узких мест

Эксперимент VI и VII

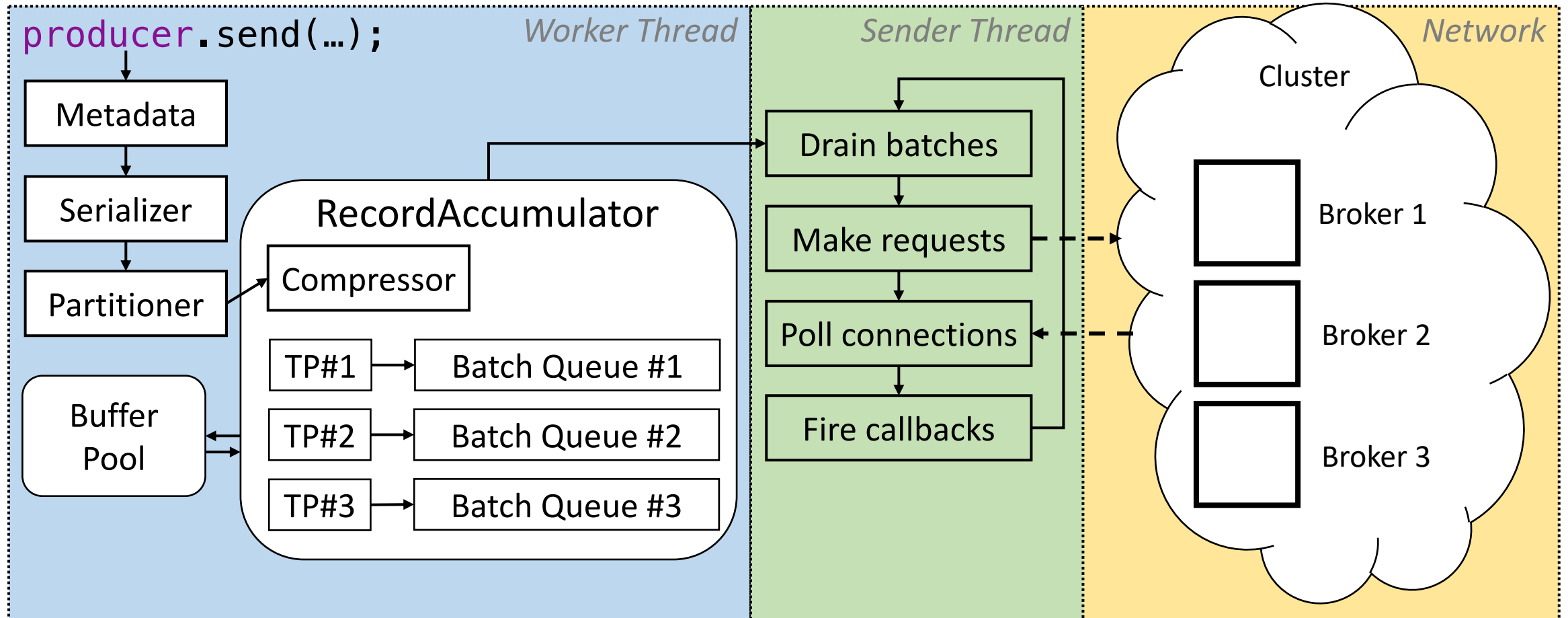
Worker Threads	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
1	343 500 $\pm$ 6 100	2.4 $\pm$ 0.2 $\mu s$	29.0 $\pm$ 4.7 ms
2			
3			

# Поиск узких мест

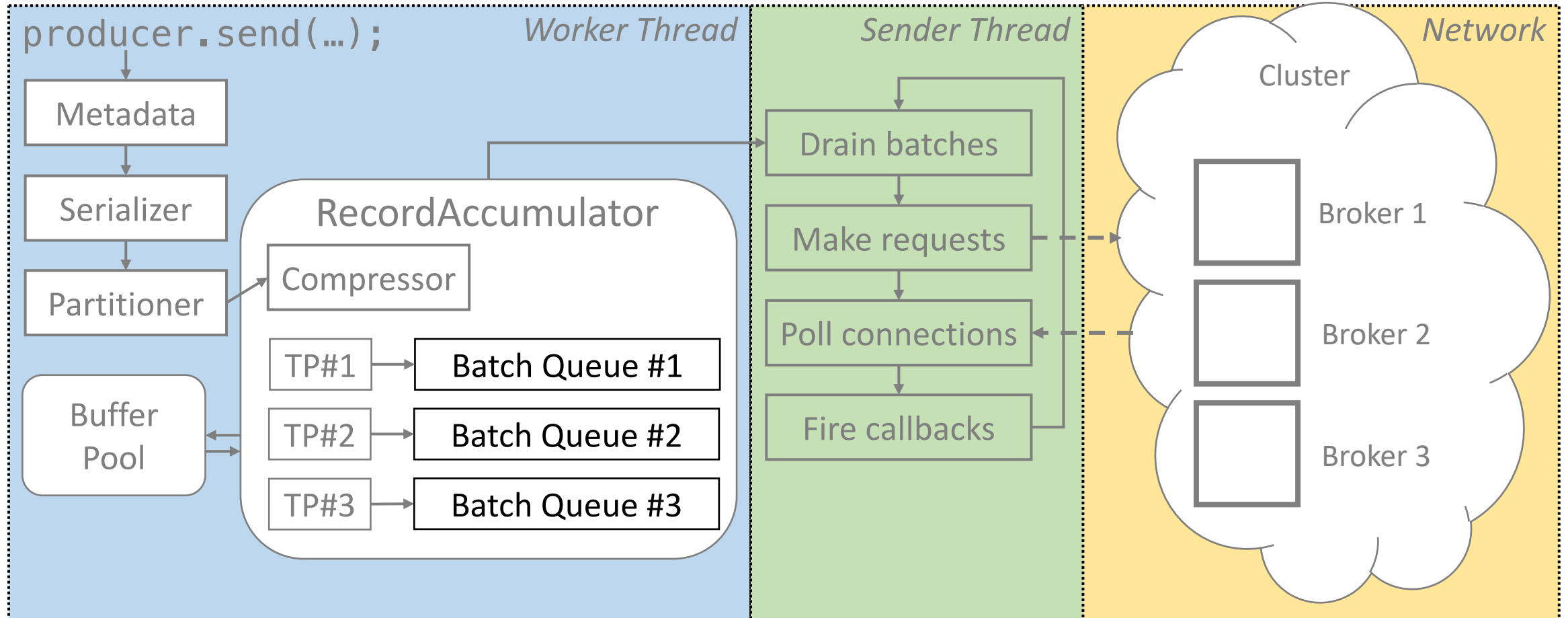
## Эксперимент VI и VII

Worker Threads	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
1	343 500 $\pm$ 6 100	2.4 $\pm$ 0.2 $\mu s$	29.0 $\pm$ 4.7 ms
2	414 600 $\pm$ 5 900 $\uparrow$	4.1 $\pm$ 0.4 $\mu s$ $\uparrow$	36.7 $\pm$ 6.1 ms
3	377 300 $\pm$ 3 000	6.7 $\pm$ 0.5 $\mu s$ $\uparrow$	31.4 $\pm$ 2.2 ms

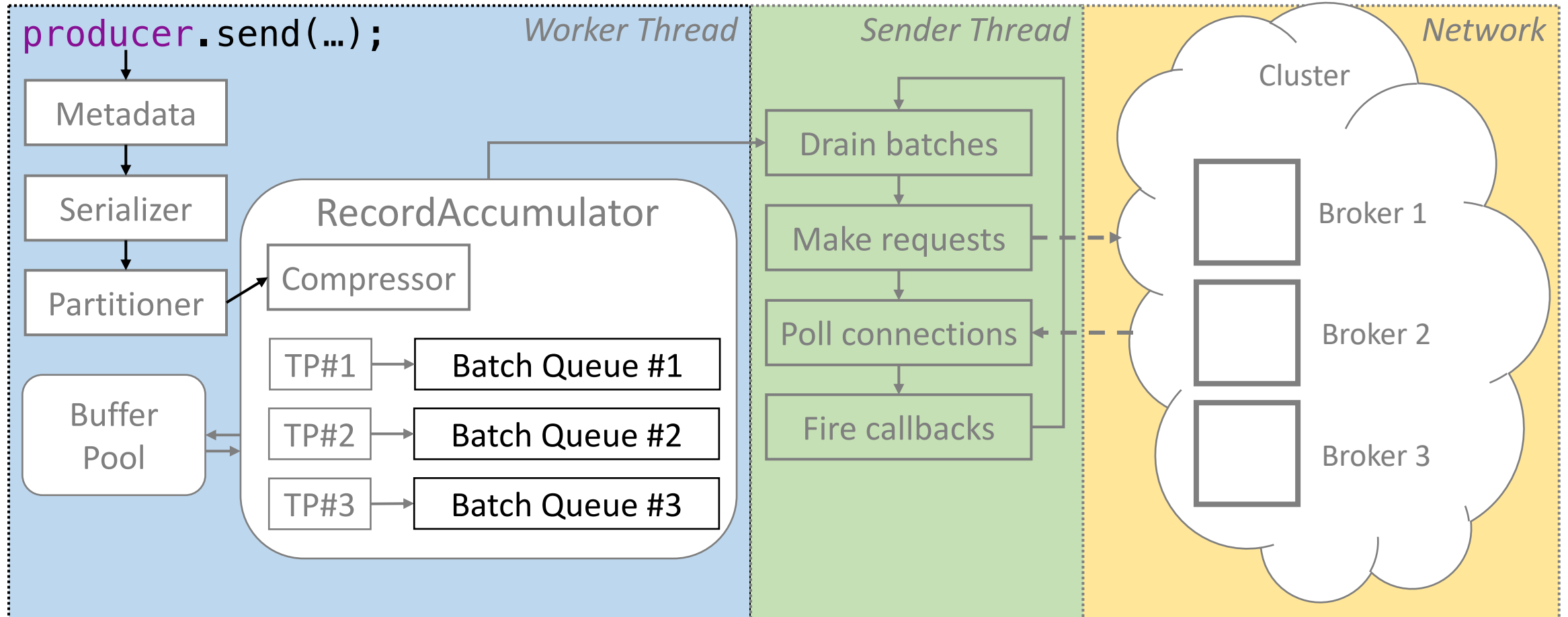
# Lock Contention



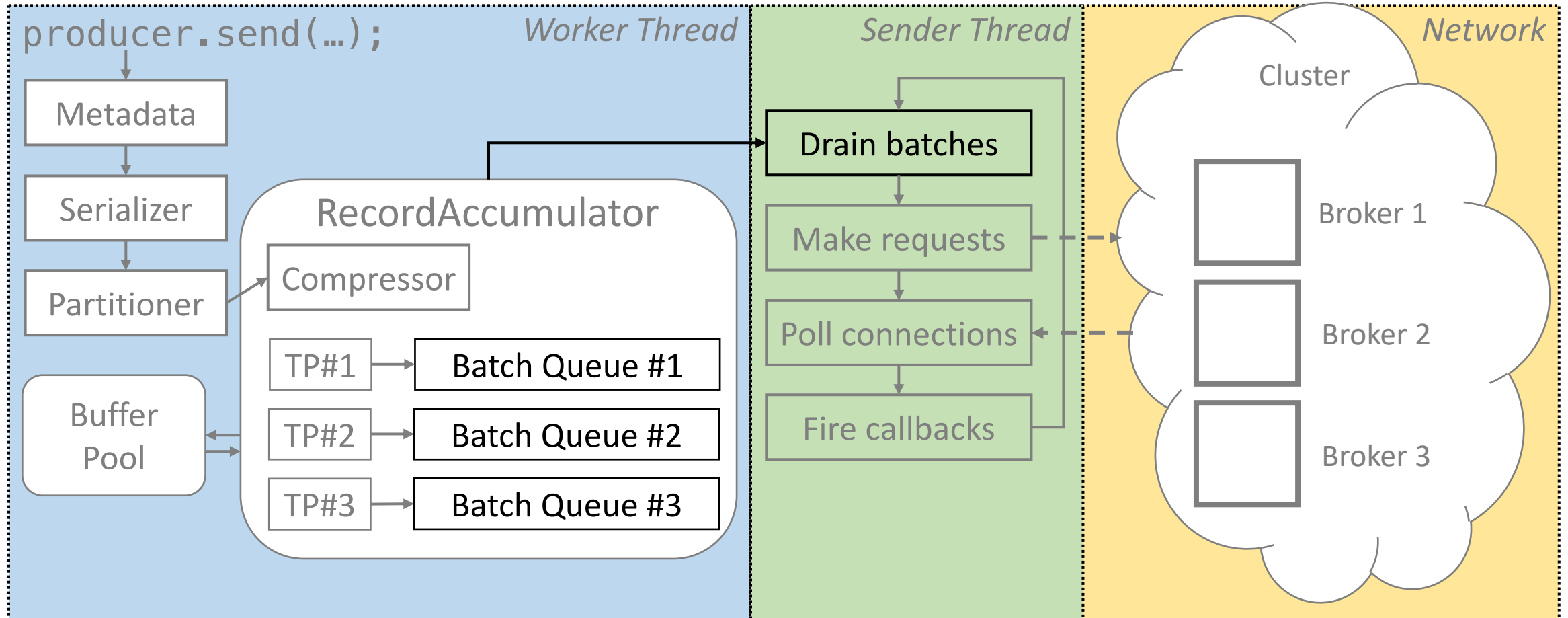
# Lock Contention



# Lock Contention



# Lock Contention



# Поиск узких мест

Эксперимент VIII, IX и X

Партиция – единица блокировки

⇒ Увеличим их количество



# Поиск узких мест

Эксперимент VIII, IX и X

Partitions	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
6	377 300 $\pm$ 3 000	6.7 $\pm$ 0.5 $\mu s$	31.4 $\pm$ 2.2 ms
9			
12			
18			

# Поиск узких мест

Эксперимент VIII, IX и X

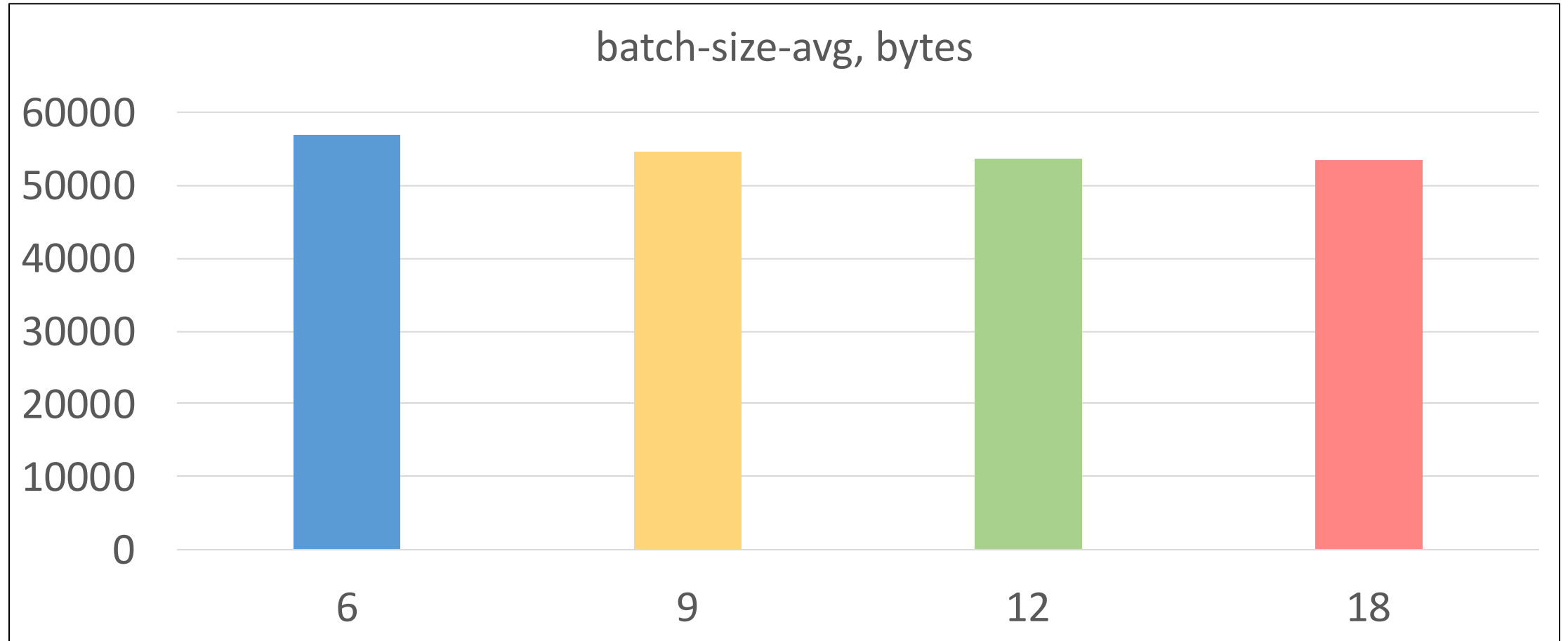
Partitions	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
6	377 300 $\pm$ 3 000	6.7 $\pm$ 0.5 $\mu s$	31.4 $\pm$ 2.2 ms
9	394 900 $\pm$ 4 100	6.3 $\pm$ 0.9 $\mu s$	23.7 $\pm$ 1.7 ms
12	397 800 $\pm$ 8 200	6.6 $\pm$ 0.7 $\mu s$	19.3 $\pm$ 2.2 ms
18	404 000 $\pm$ 7 000	6.4 $\pm$ 0.7 $\mu s$	19.8 $\pm$ 1.6 ms

# Поиск узких мест

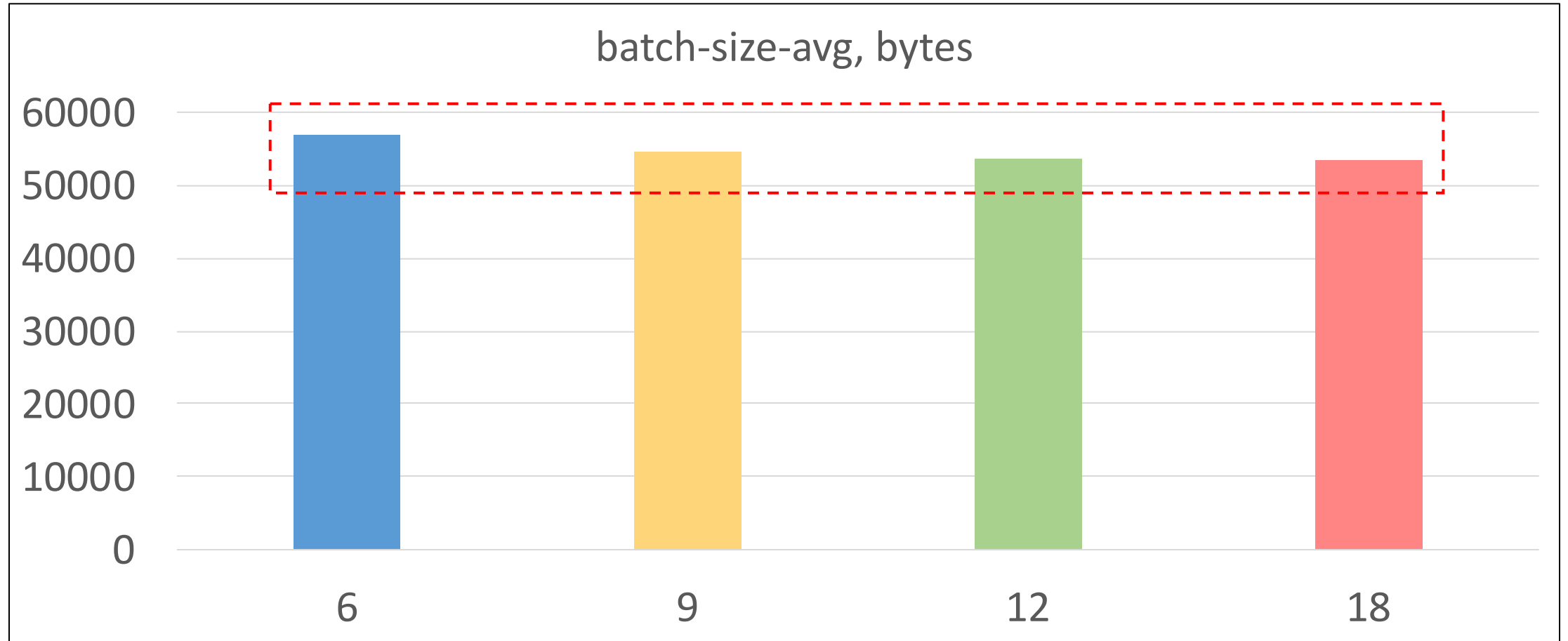
Эксперимент VIII, IX и X

Partitions	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
6	377 300 $\pm$ 3 000	6.7 $\pm$ 0.5 $\mu s$	<b>31.4 <math>\pm</math> 2.2 ms</b>
9	394 900 $\pm$ 4 100	6.3 $\pm$ 0.9 $\mu s$	<b>23.7 <math>\pm</math> 1.7 ms</b>
12	397 800 $\pm$ 8 200	6.6 $\pm$ 0.7 $\mu s$	<b>19.3 <math>\pm</math> 2.2 ms</b>
18	404 000 $\pm$ 7 000	6.4 $\pm$ 0.7 $\mu s$	<b>19.8 <math>\pm</math> 1.6 ms</b>

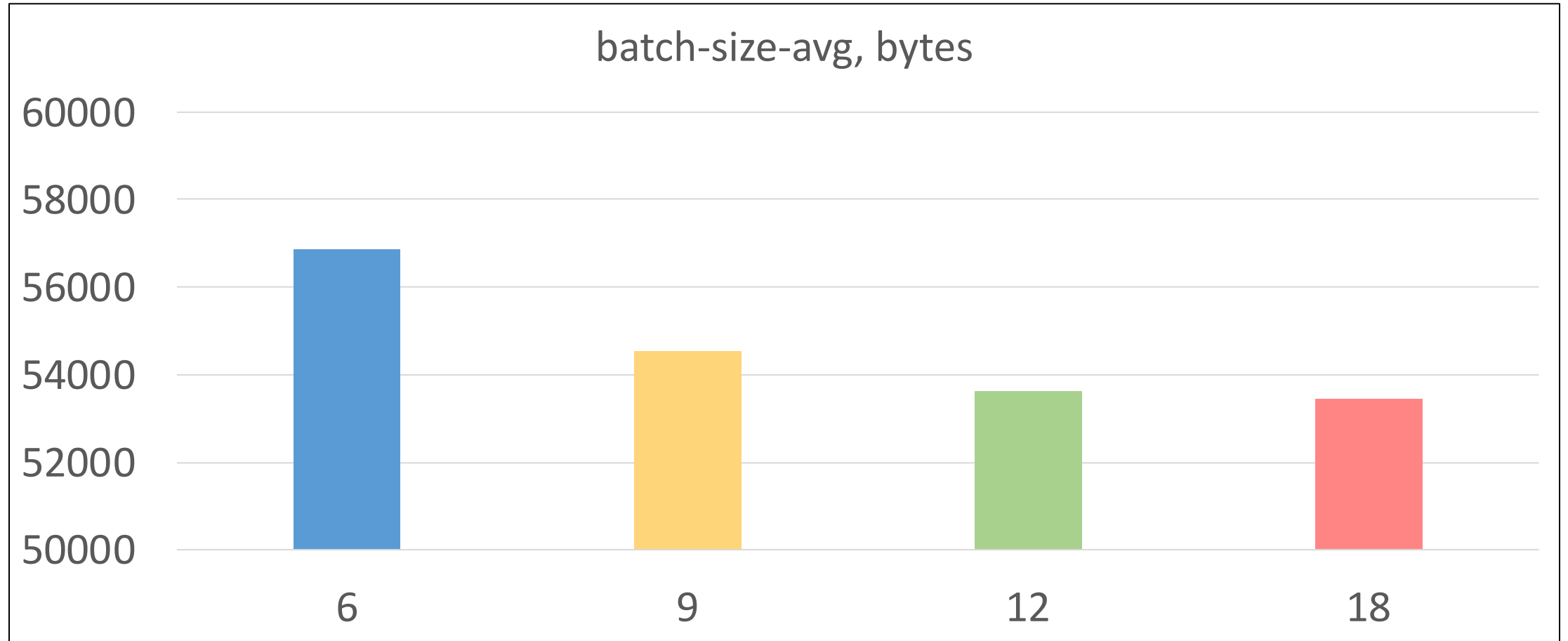
# Поиск узких мест



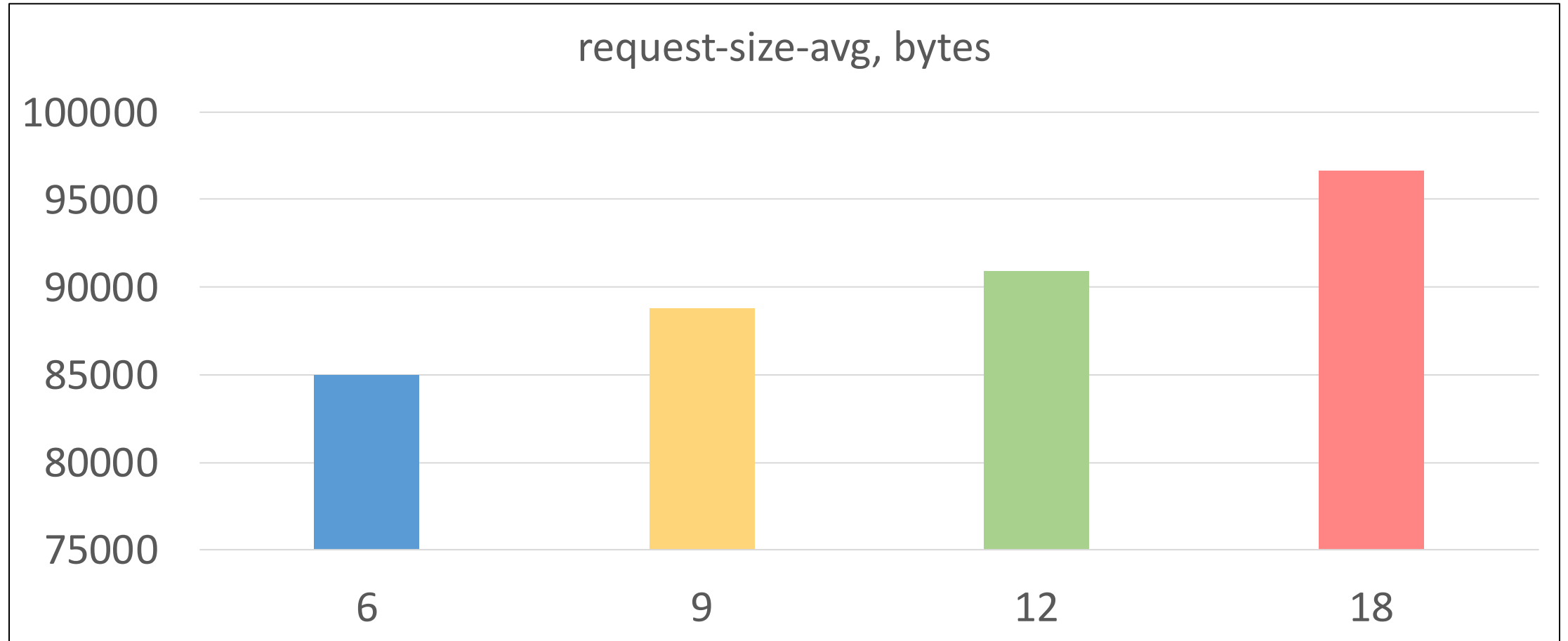
# Поиск узких мест



# Поиск узких мест



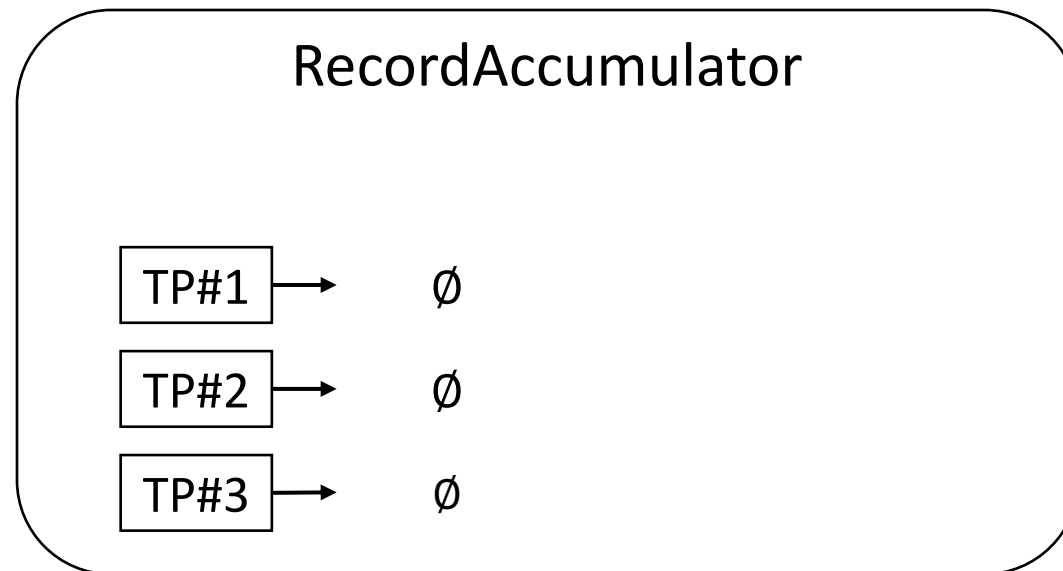
# Поиск узких мест



# KIP-480

До версии 2.4

- партиция выбирается по Round-robin

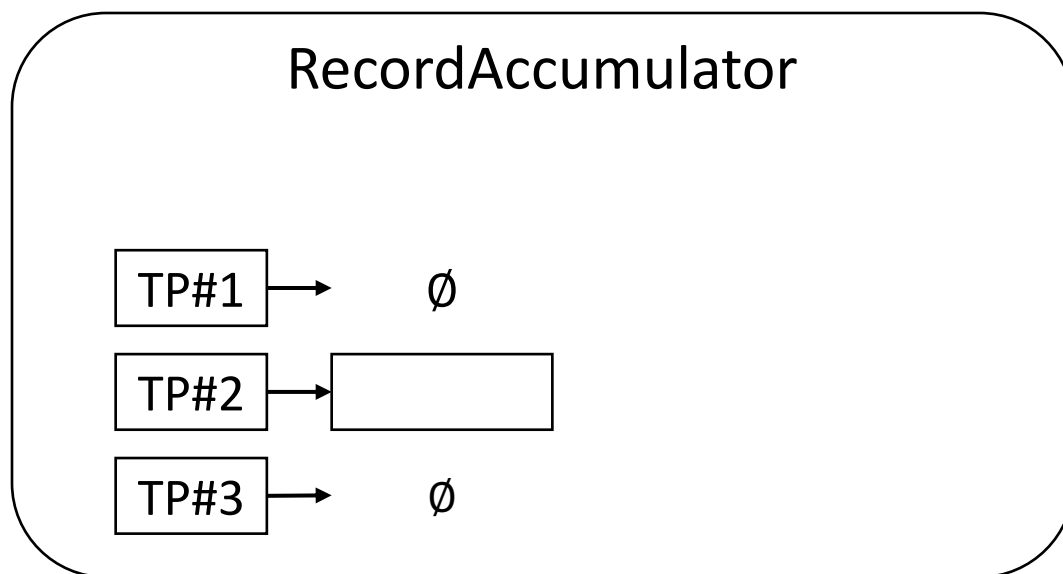




# KIP-480

До версии 2.4

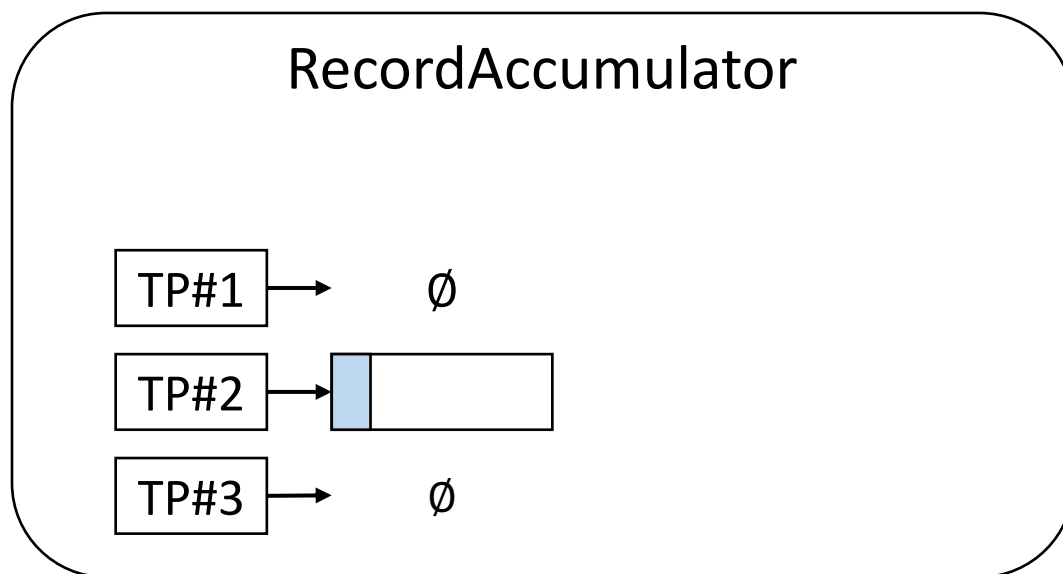
- партиция выбирается по Round-robin



# KIP-480

До версии 2.4

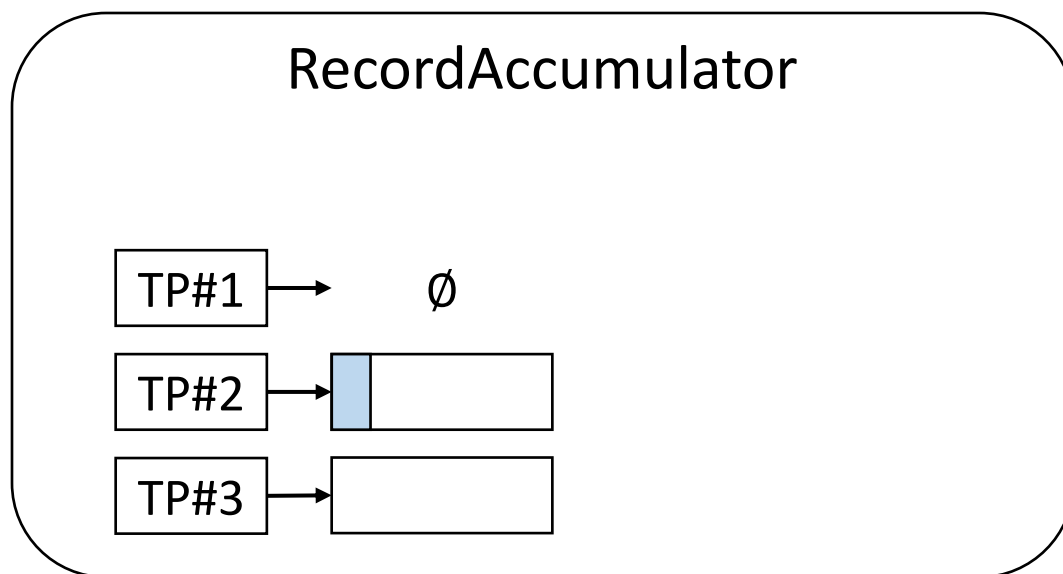
- партиция выбирается по Round-robin



# KIP-480

До версии 2.4

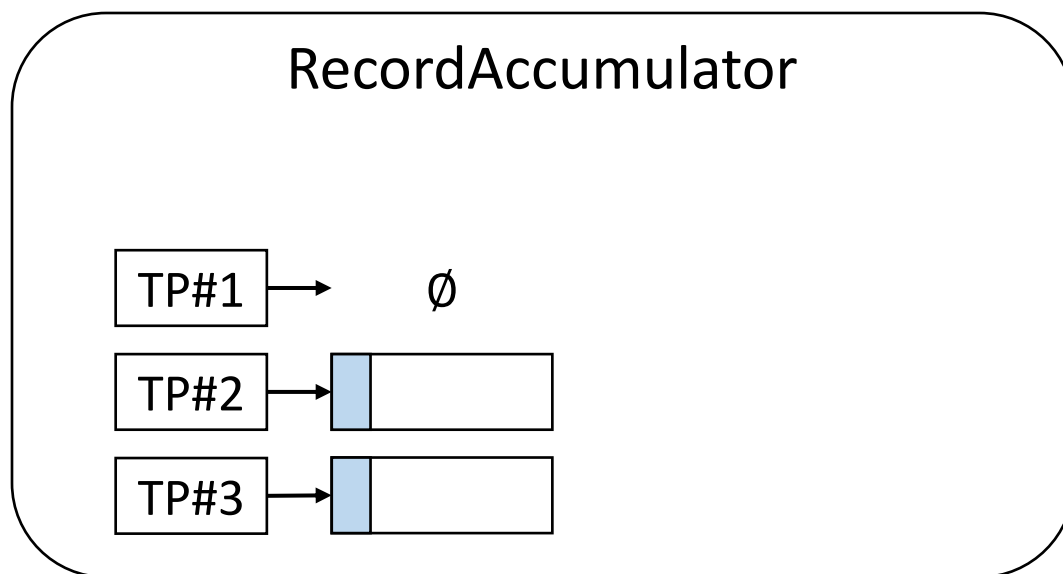
- партиция выбирается по Round-robin



# KIP-480

До версии 2.4

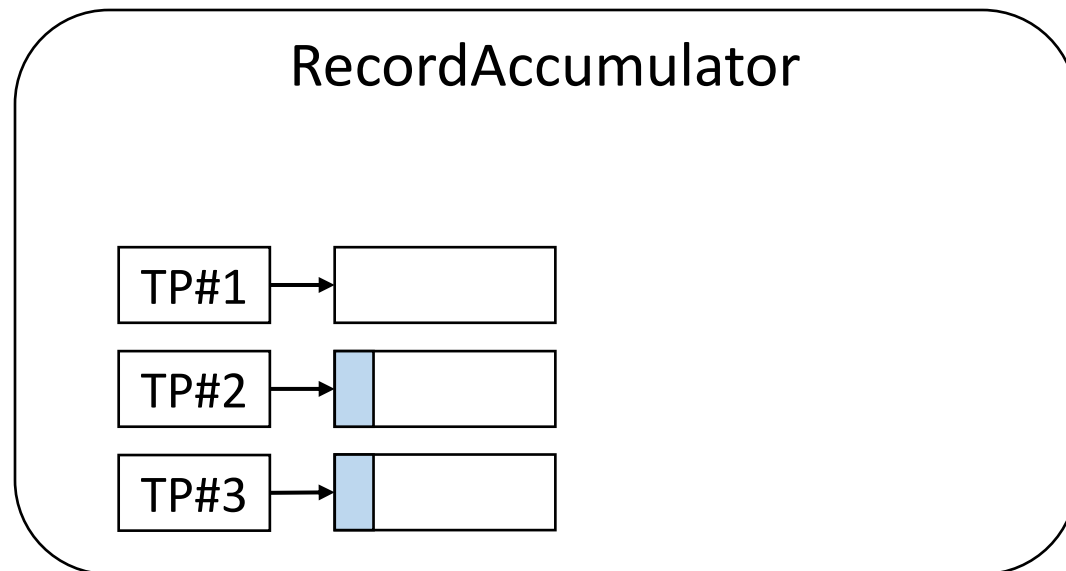
- партиция выбирается по Round-robin



# KIP-480

До версии 2.4

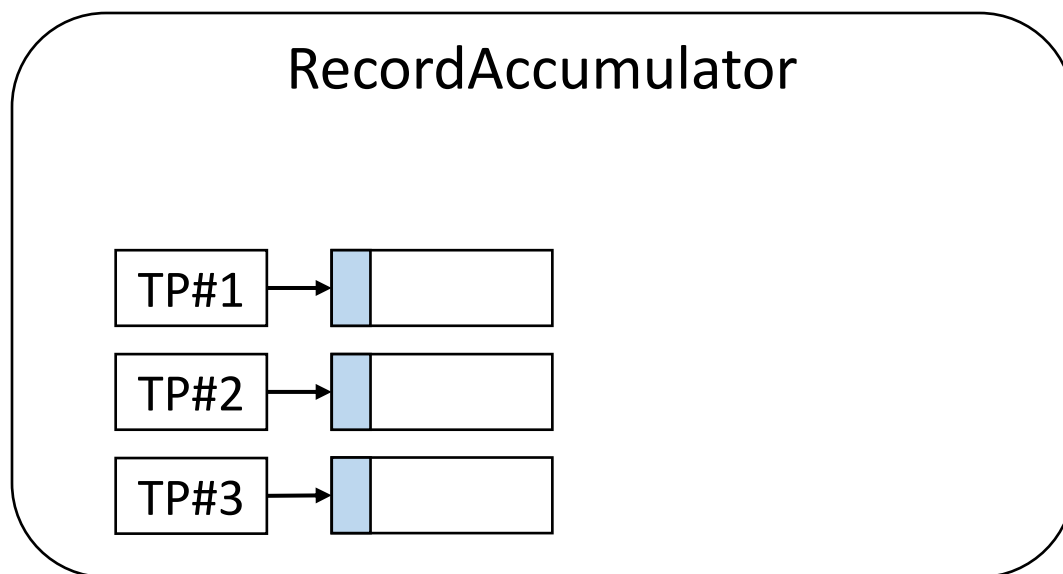
- партиция выбирается по Round-robin



# KIP-480

До версии 2.4

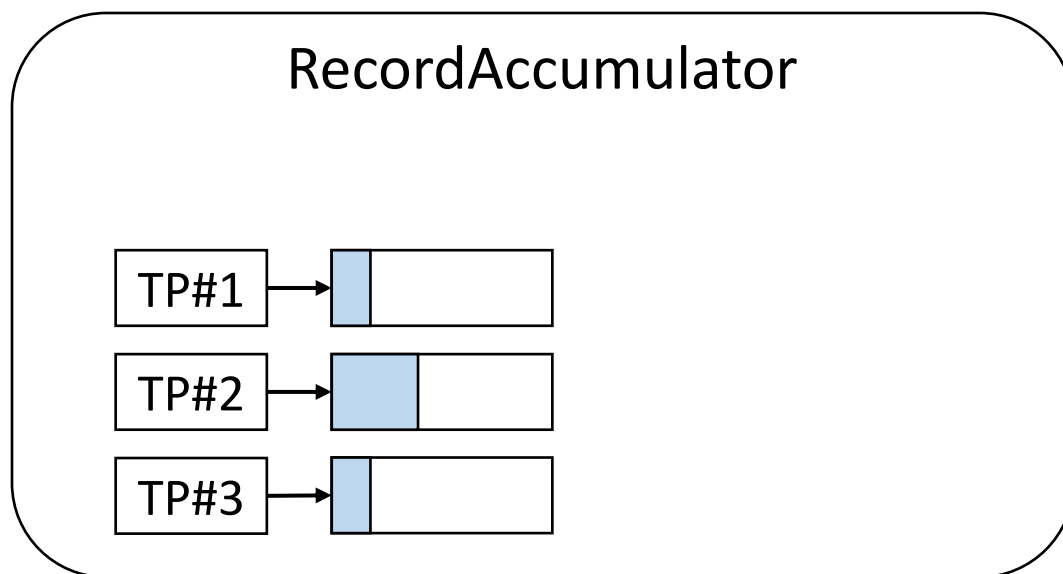
- партиция выбирается по Round-robin



# KIP-480

До версии 2.4

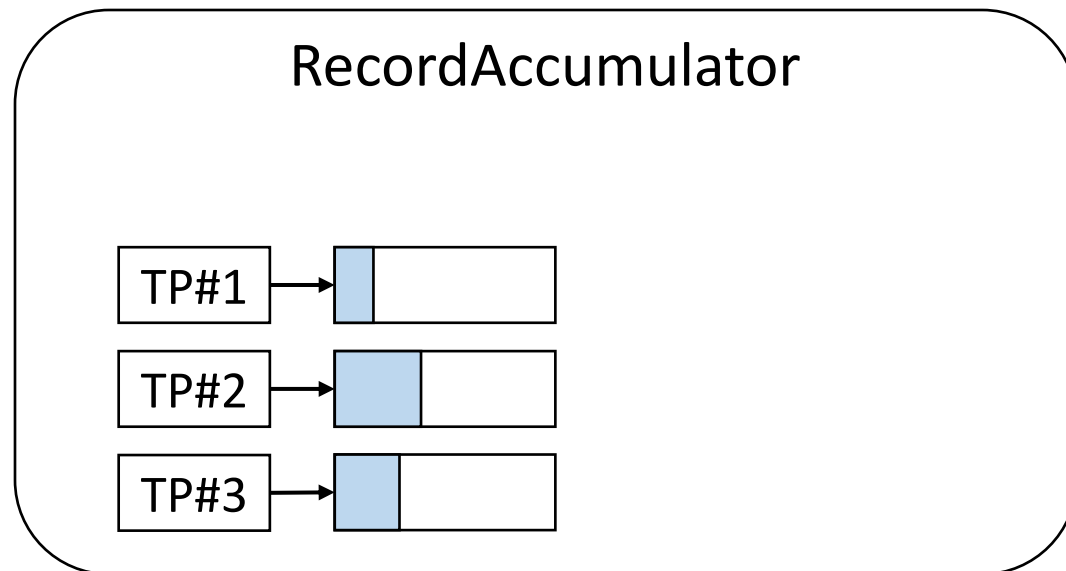
- партиция выбирается по Round-robin



# KIP-480

До версии 2.4

- партиция выбирается по Round-robin

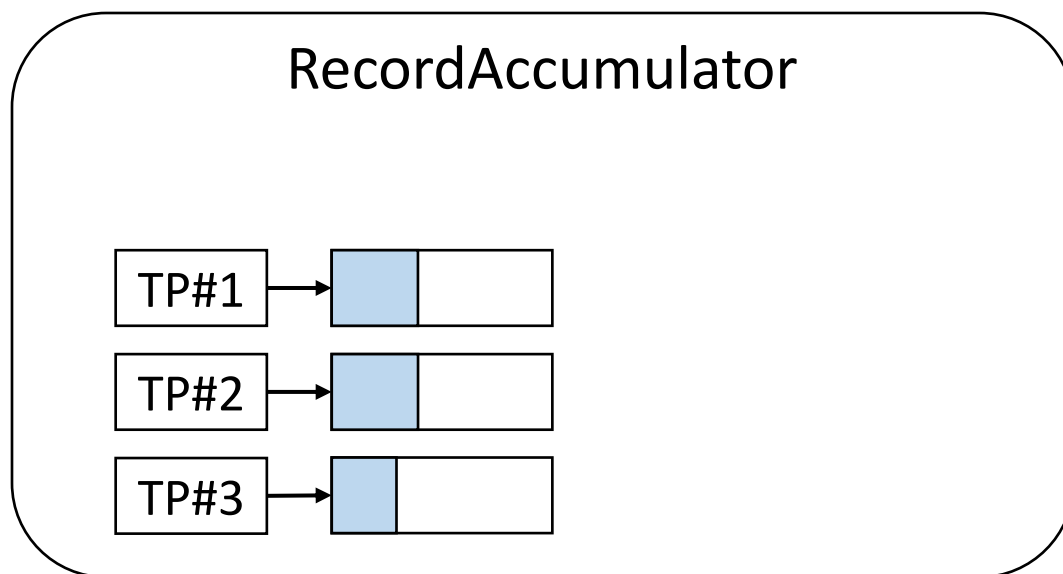




# KIP-480

До версии 2.4

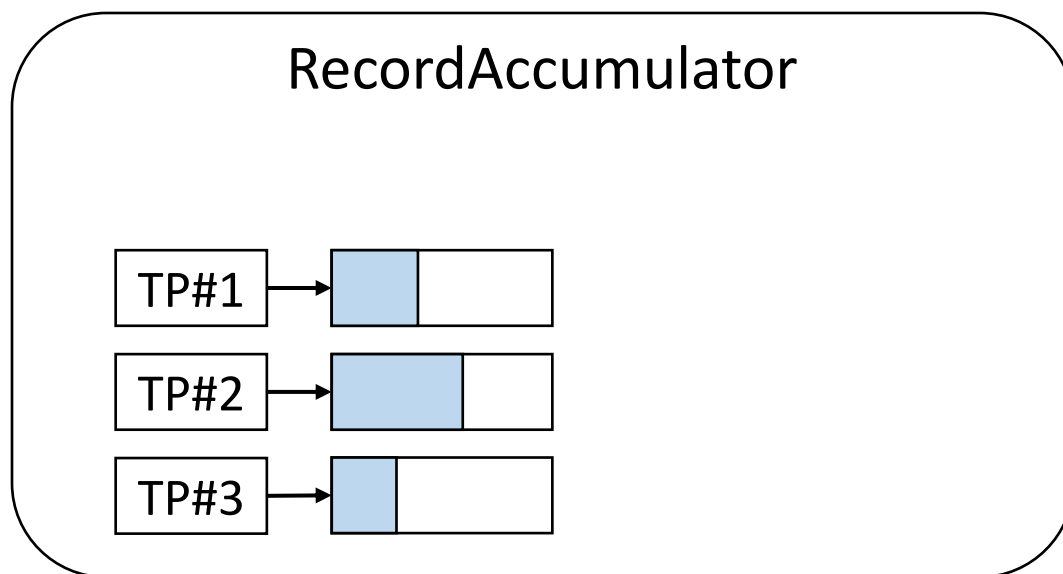
- партиция выбирается по Round-robin



# KIP-480

До версии 2.4

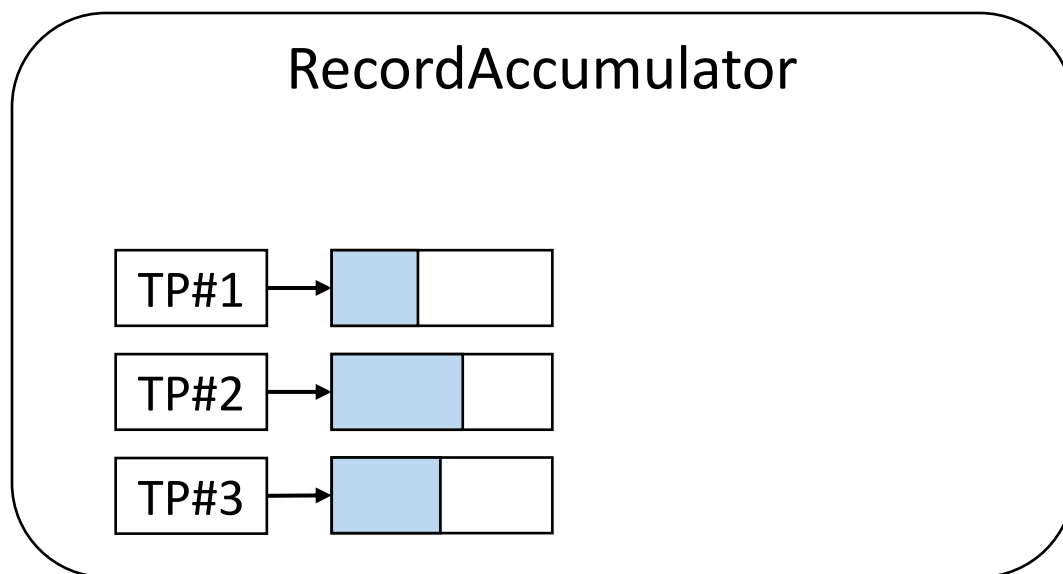
- партиция выбирается по Round-robin



# KIP-480

До версии 2.4

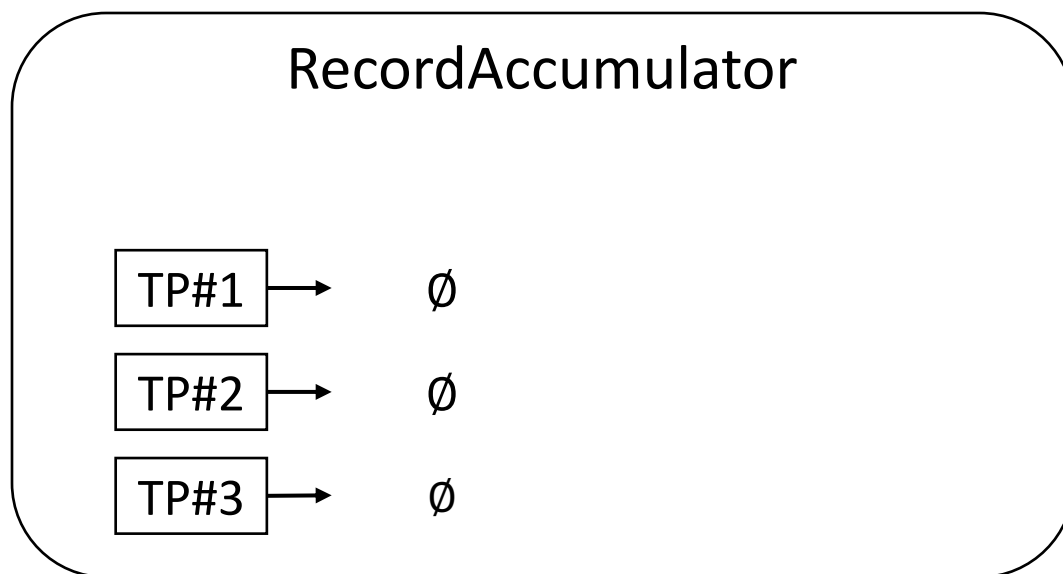
- партиция выбирается по Round-robin



# KIP-480

С версии 2.4

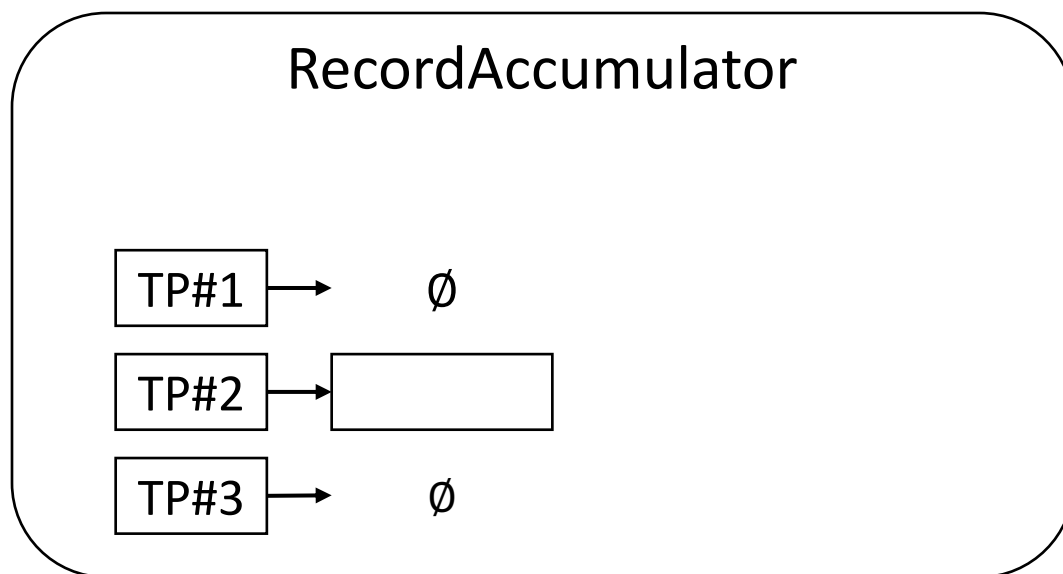
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

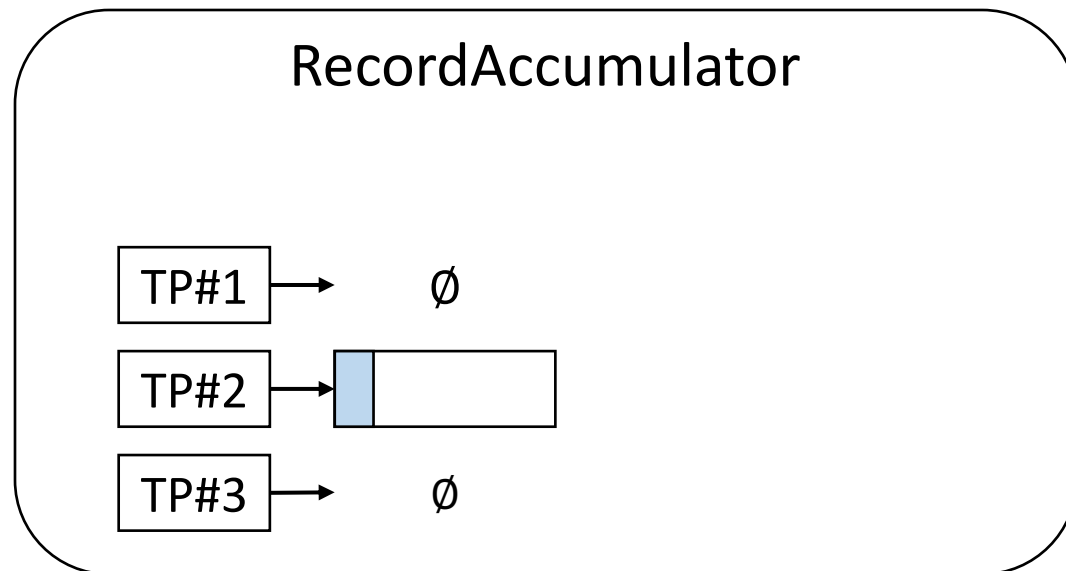
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

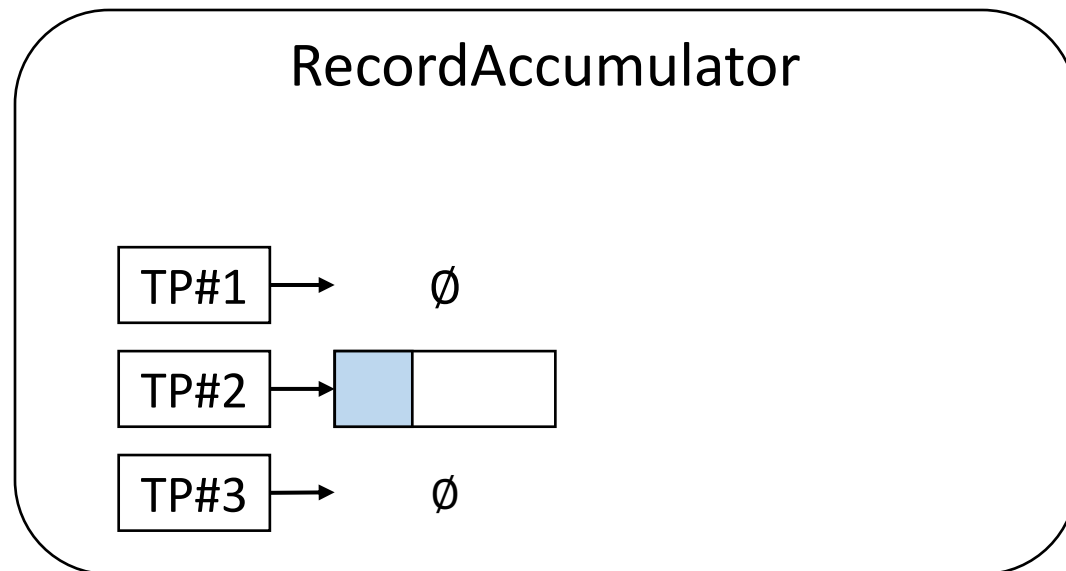
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

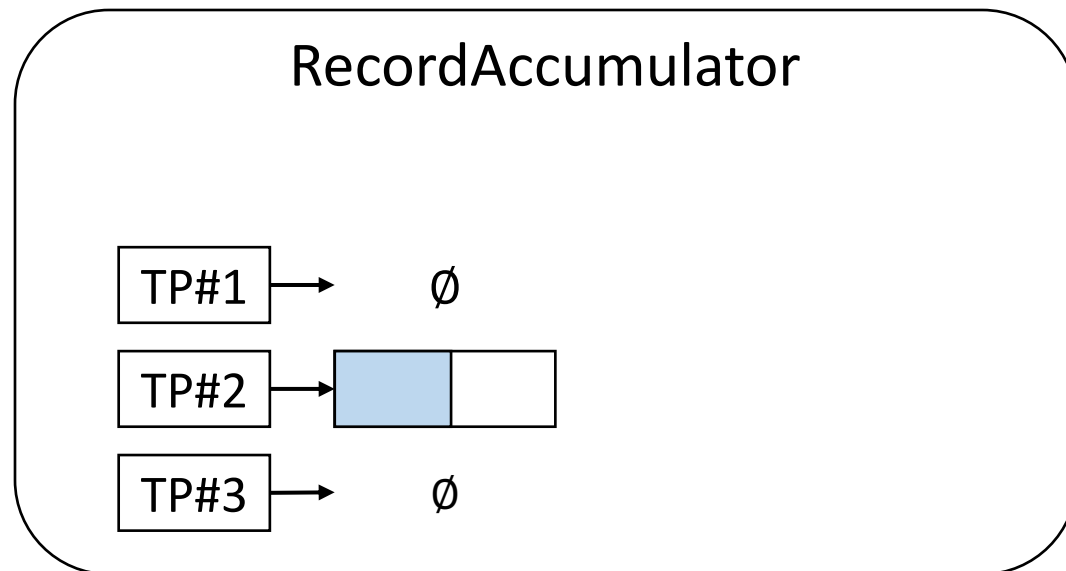
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом

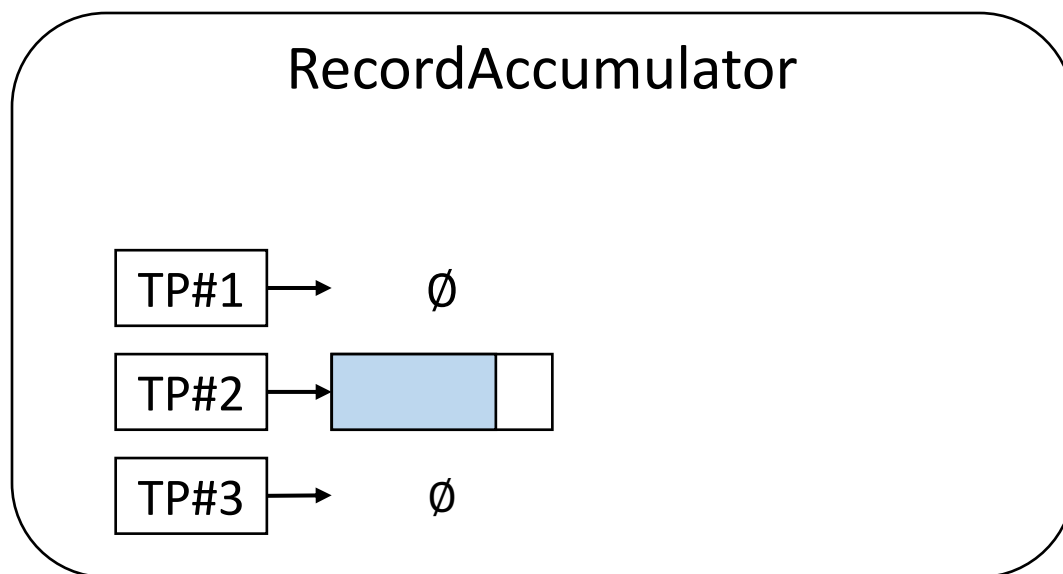




# KIP-480

С версии 2.4

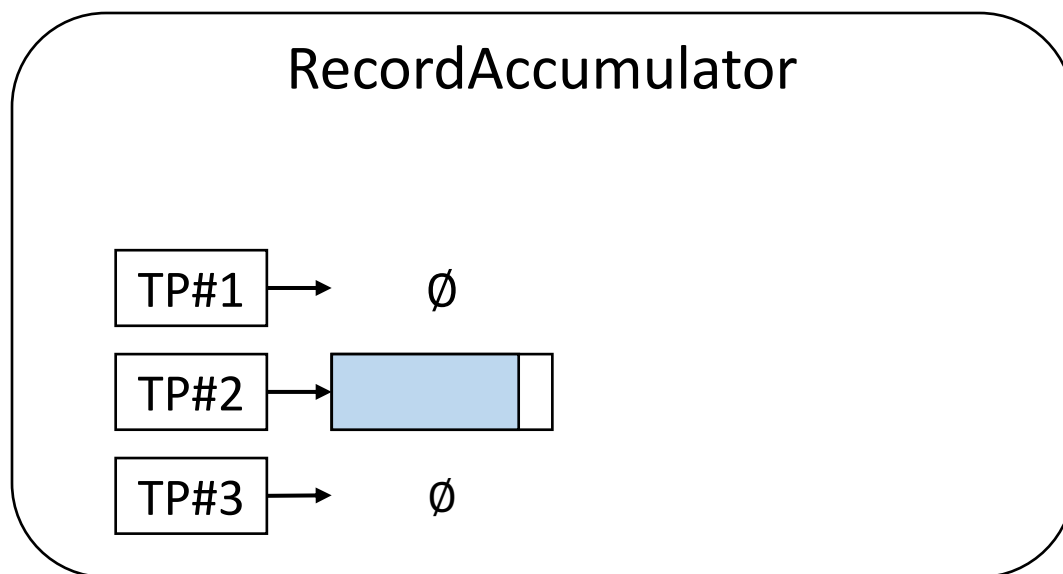
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

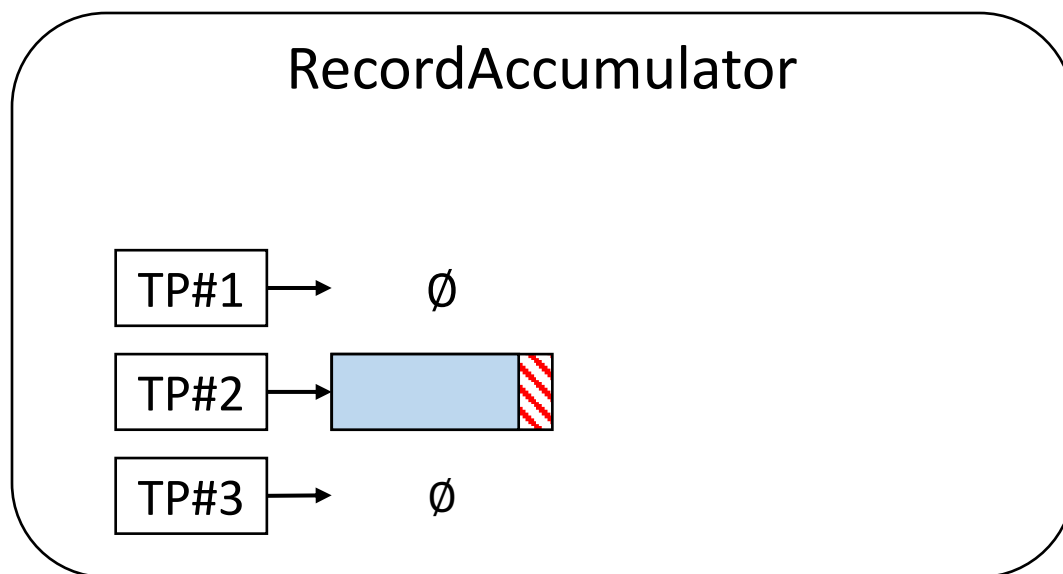
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

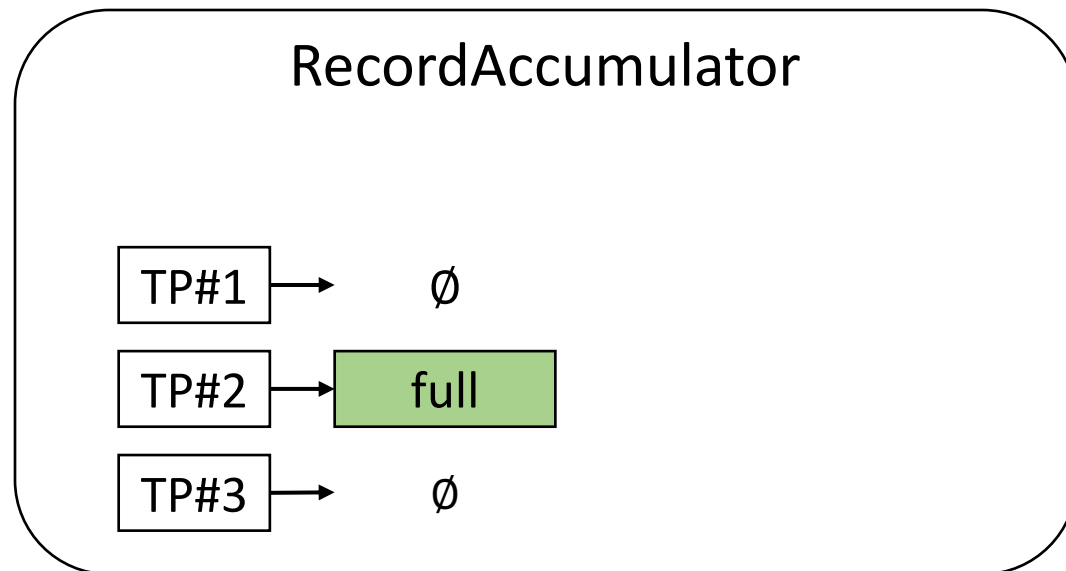
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

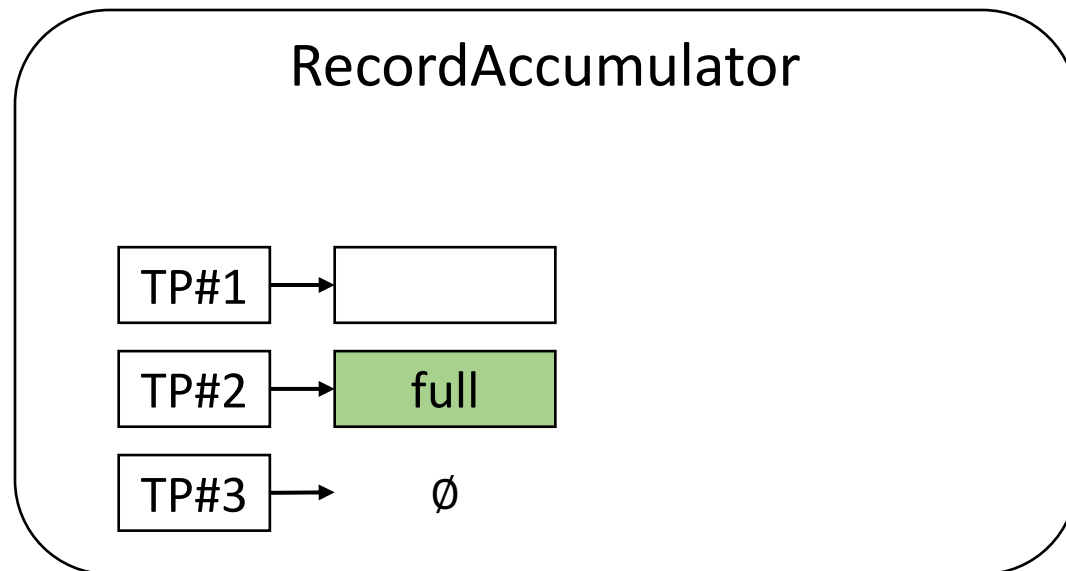
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

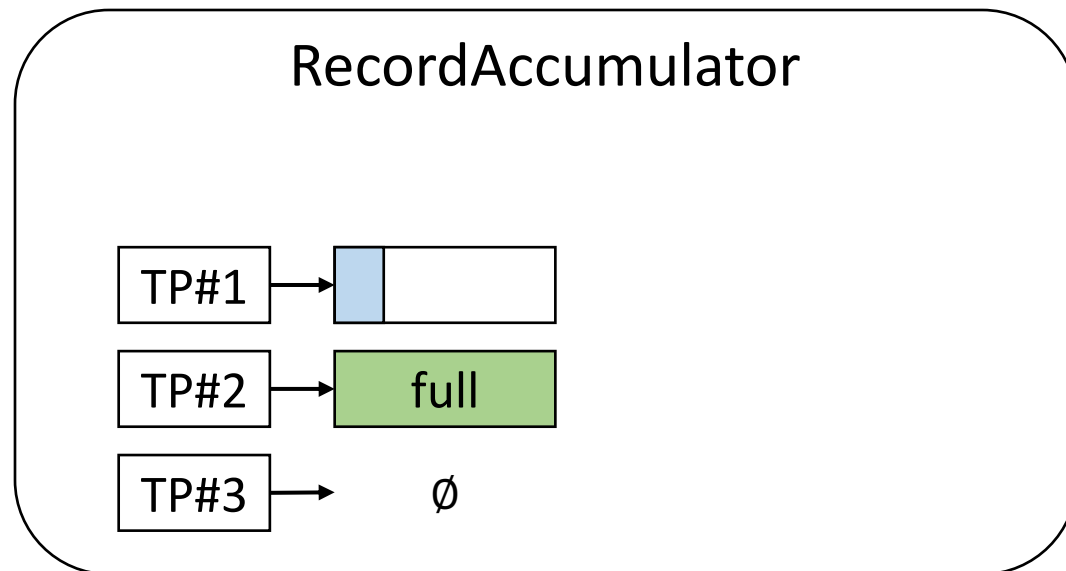
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

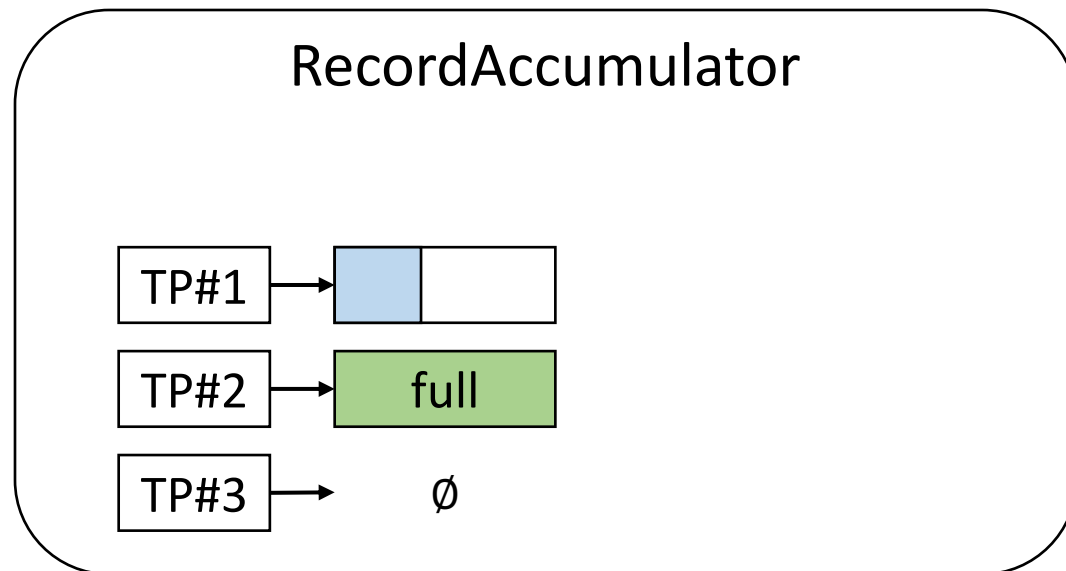
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

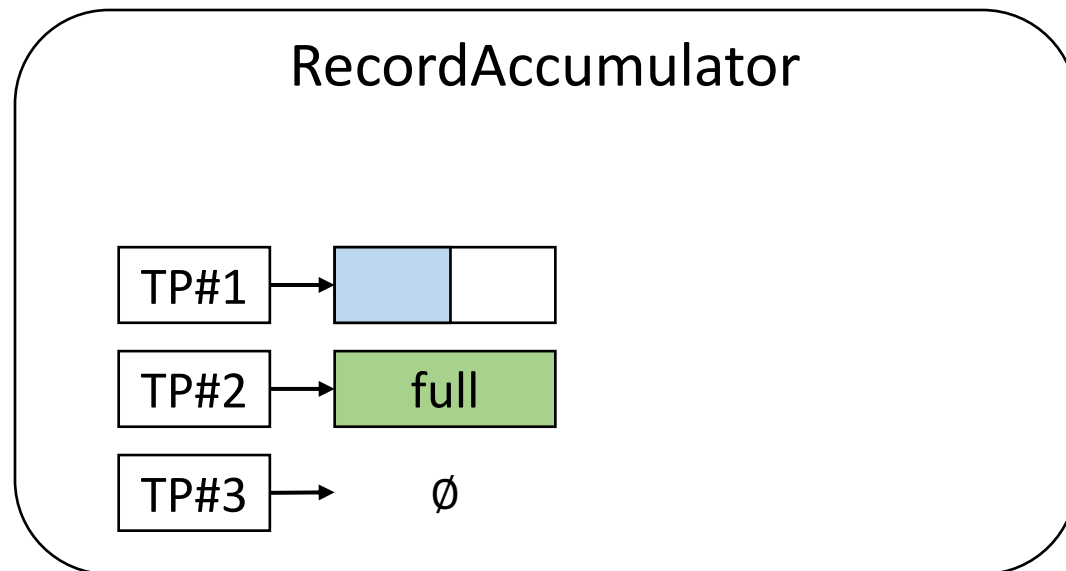
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



# KIP-480

С версии 2.4

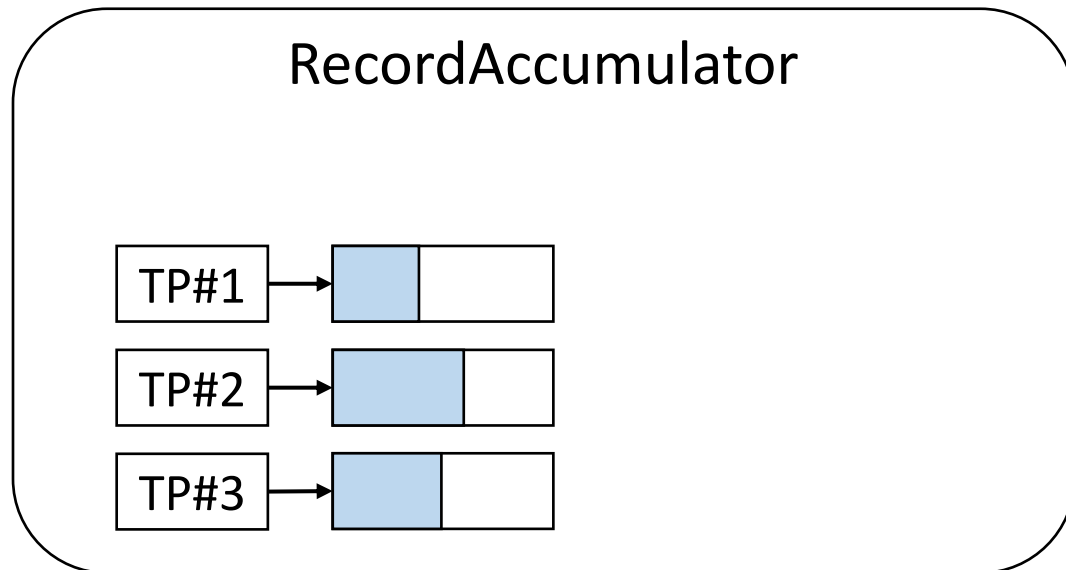
- партиция выбирается одна, пока не заполнится пачка
- следующая партиция выбирается случайным образом



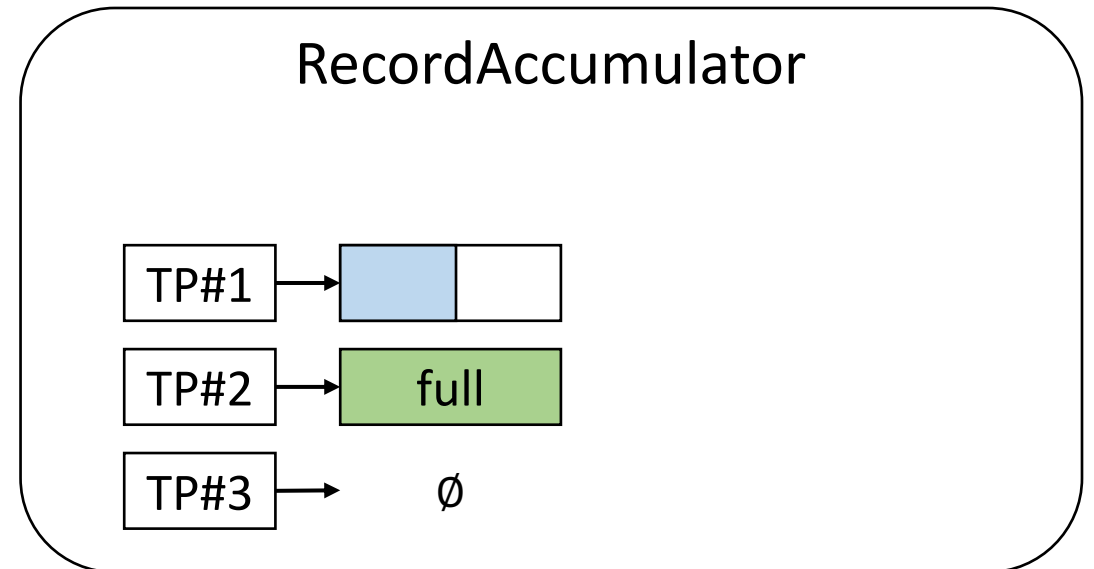


# KIP-480

До версии 2.4



С версии 2.4




# KIP-480

Пачки накапливаются быстрее  $\Rightarrow$

# KIP-480

Пачки накапливаются быстрее  $\Rightarrow$

1. Можно не ждать `linger.ms`  $\Rightarrow$  `record-queue-time-avg` 

# KIP-480

Пачки накапливаются быстрее  $\Rightarrow$

1. Можно не ждать `linger.ms`  $\Rightarrow$  `record-queue-time-avg`  $\downarrow$
2. Больше запросов  $\Rightarrow$  `request-latency-avg`  $\uparrow$

# KIP-480

Пачки накапливаются быстрее  $\Rightarrow$

1. Можно не ждать `linger.ms`  $\Rightarrow$  `record-queue-time-avg`  $\downarrow$
2. Больше запросов  $\Rightarrow$  `request-latency-avg`  $\uparrow$

$$\text{Sender Latency} = (\text{record-queue-time-avg} / 2) + \text{request-latency-avg}$$

# Поиск узких мест

Эксперимент XI

Разные версии клиента

Version	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
2.2.0			
2.4.0	404 000 $\pm$ 7 000	6.4 $\pm$ 0.7 $\mu s$	19.8 $\pm$ 1.6 ms

# Поиск узких мест

Эксперимент XI

Разные версии клиента

Version	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
2.2.0	421 600 $\pm$ 3 000 $\uparrow$	6.2 $\pm$ 0.7 $\mu s$	16.7 $\pm$ 0.6 ms $\downarrow$
2.4.0	404 000 $\pm$ 7 000	6.4 $\pm$ 0.7 $\mu s$	19.8 $\pm$ 1.6 ms

# Поиск узких мест

Эксперимент XI

Разные версии клиента

Version	Throughput, rps	Worker Latency, $\mu\text{s}$	Sender Latency, ms
2.2.0	421 600 $\pm$ 3 000 $\uparrow$	6.2 $\pm$ 0.7 $\mu\text{s}$	16.7 $\pm$ 0.6 ms $\downarrow$
2.4.0	404 000 $\pm$ 7 000	6.4 $\pm$ 0.7 $\mu\text{s}$	19.8 $\pm$ 1.6 ms

Зависит от распределения лидерства!



# Поиск узких мест

## Эксперимент XII

- Будем указывать партицию явно
- У различных Worker Thread будут свои номера партиций

# Поиск узких мест

## Эксперимент XII

Partitioner	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
Round-robin (2.2)	421 600 $\pm$ 3 000	6.2 $\pm$ 0.7 $\mu s$	16.7 $\pm$ 0.6 ms
Sticky (2.4)	404 000 $\pm$ 7 000	6.4 $\pm$ 0.7 $\mu s$	19.8 $\pm$ 1.6 ms
No contention			

# Поиск узких мест

## Эксперимент XII

Partitioner	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
Round-robin (2.2)	421 600 $\pm$ 3 000	6.2 $\pm$ 0.7 $\mu s$	16.7 $\pm$ 0.6 ms
Sticky (2.4)	404 000 $\pm$ 7 000	6.4 $\pm$ 0.7 $\mu s$	19.8 $\pm$ 1.6 ms
No contention	500 000 $\pm$ 400 $\uparrow$	4.5 $\pm$ 1.1 $\mu s$ $\downarrow$	23.5 $\pm$ 1.8 ms $\uparrow$

# Поиск узких мест

Эксперимент XIII, XIV и XV

- Уберём ограничение на RPS
- И докинем потоков

# Поиск узких мест

Эксперимент XIII, XIV и XV

Worker Threads	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
3			
6			
9			

# Поиск узких мест

Эксперимент XIII, XIV и XV

Worker Threads	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
3	610 700 $\pm$ 10 100	3.6 $\pm$ 0.4 $\mu s$	189 $\pm$ 221 ms
6			
9			


# Поиск узких мест

Эксперимент XIII, XIV и XV

Worker Threads	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
3	610 700 $\pm$ 10 100	3.6 $\pm$ 0.4 $\mu s$	<del>180 <math>\pm</math> 221 ms</del>
6			
9			

# Поиск узких мест

Эксперимент XIII, XIV и XV

Worker Threads	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
3	610 700 $\pm$ 10 100	3.6 $\pm$ 0.4 $\mu s$	<del>187 <math>\pm</math> 221 ms</del>
6	586 700 $\pm$ 21 800	8.8 $\pm$ 0.6 $\mu s$ 	214 $\pm$ 239 ms
9			



# Поиск узких мест

Эксперимент XIII, XIV и XV

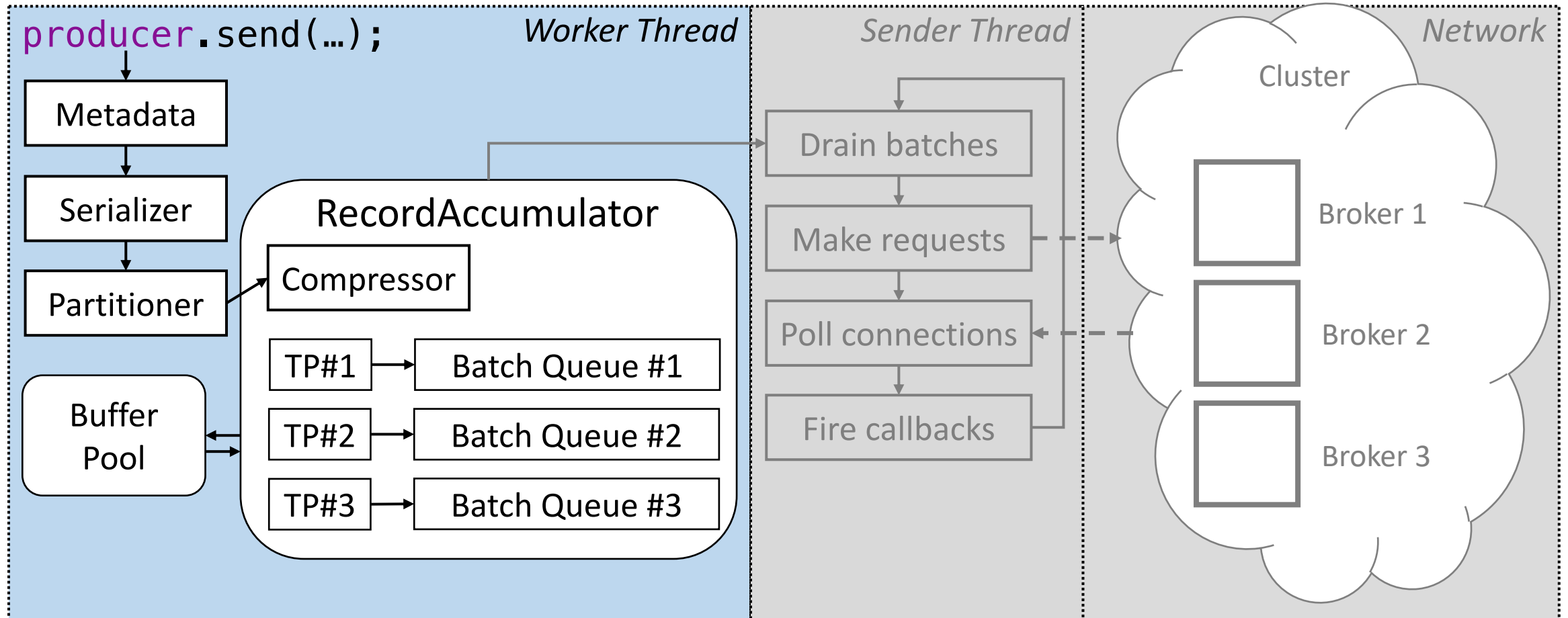
Worker Threads	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
3	610 700 $\pm$ 10 100	3.6 $\pm$ 0.4 $\mu s$	<del>180 <math>\pm</math> 221 ms</del>
6	586 700 $\pm$ 21 800	8.8 $\pm$ 0.6 $\mu s$ $\uparrow$	<del>21 <math>\pm</math> 199 ms</del>
9			

# Поиск узких мест

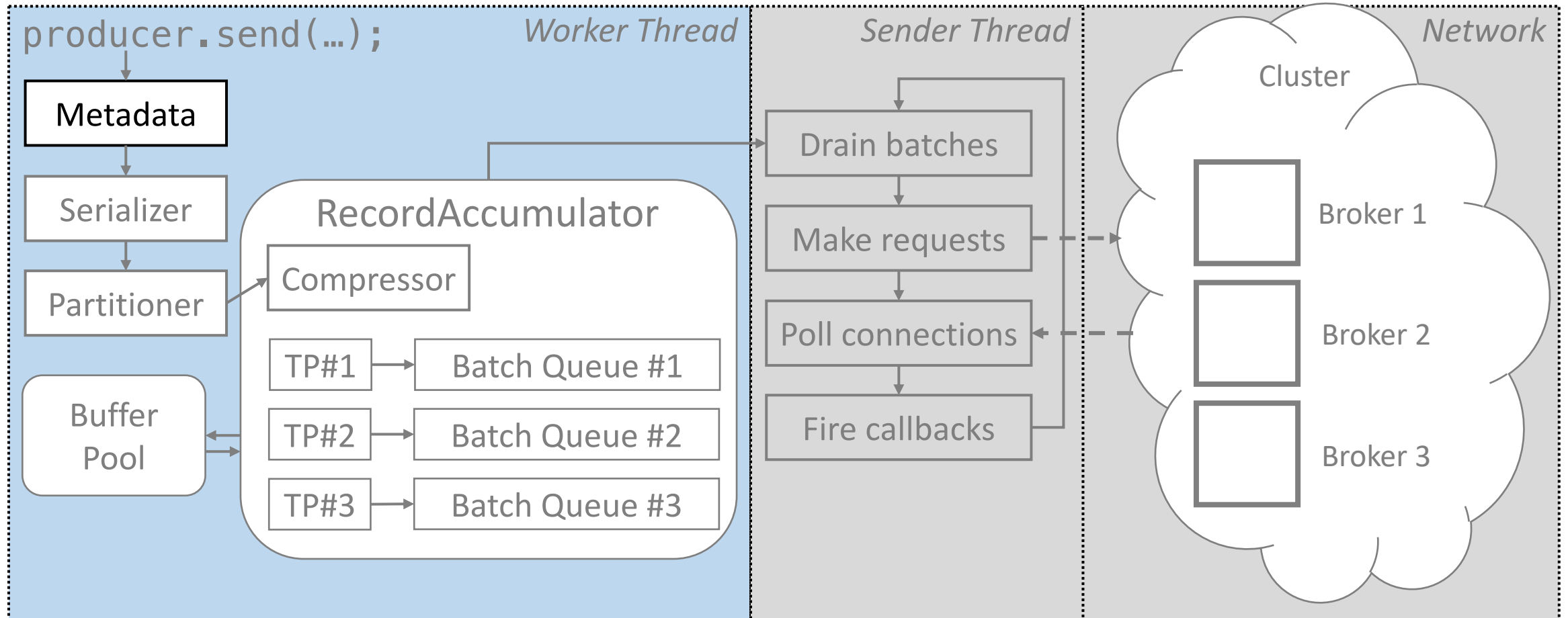
Эксперимент XIII, XIV и XV

Worker Threads	Throughput, rps	Worker Latency, $\mu s$	Sender Latency, ms
3	610 700 $\pm$ 10 100	3.6 $\pm$ 0.4 $\mu s$	<del>180 <math>\pm</math> 221 ms</del>
6	586 700 $\pm$ 21 800	8.8 $\pm$ 0.6 $\mu s$ ↑	<del>21 <math>\pm</math> 199 ms</del>
9	551 800 $\pm$ 22 900	15.2 $\pm$ 1.1 $\mu s$ ↑↑	36 $\pm$ 11 ms

# Поиск узких мест



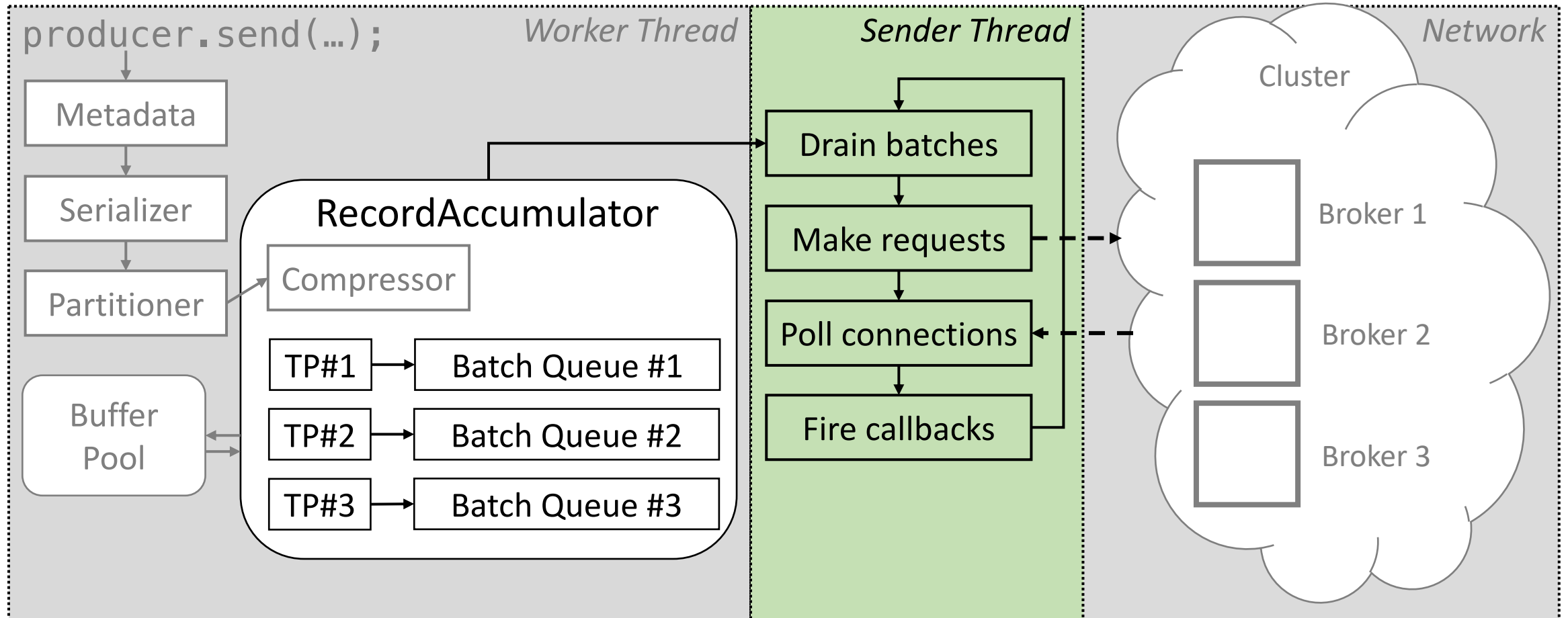
# Поиск узких мест



# Поиск узких мест

```
class ProducerMetadata extends Metadata {  
    synchronized Cluster fetch() { /* ... */ }  
  
    synchronized void add(  
        String topic,  
        long nowMs) { /* ... */ }  
  
    /* ... */  
}
```

# Узкие места – Sender Thread



# Узкие места – Sender Thread

Callback Latency

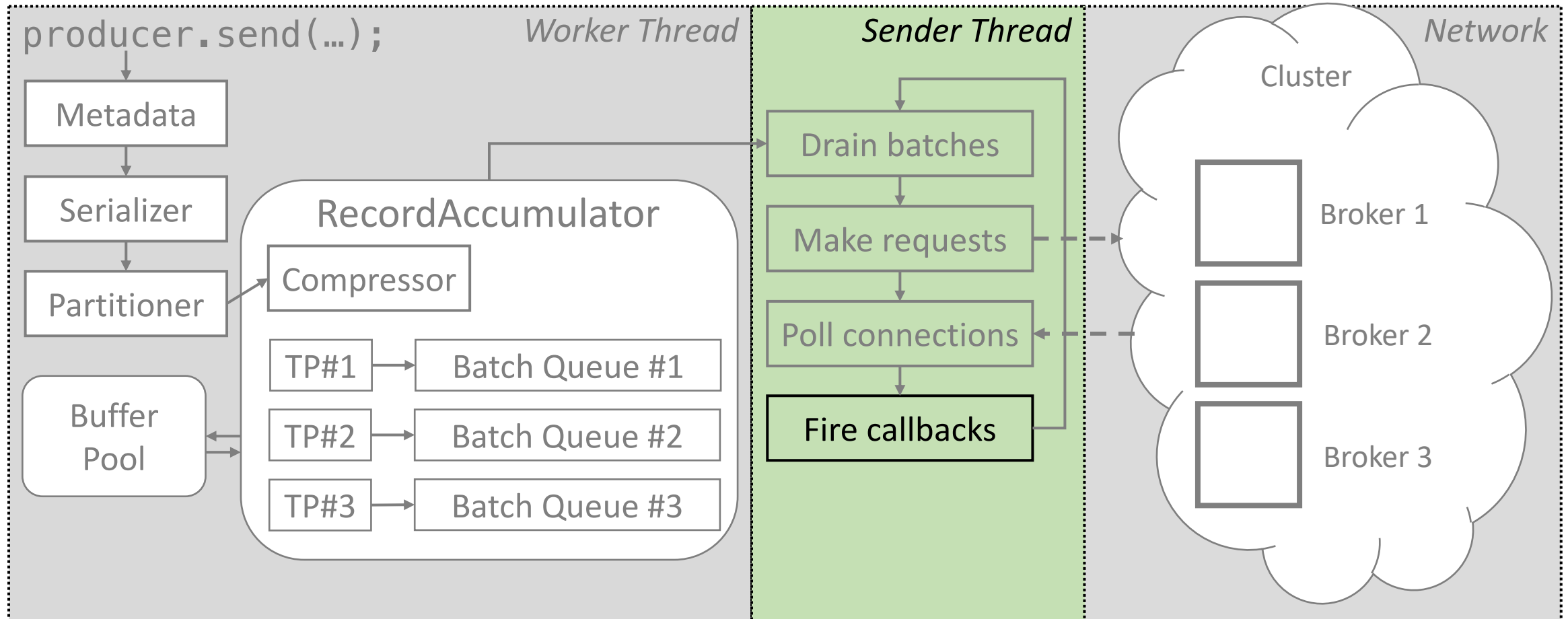
# Узкие места – Sender Thread

Callback Latency

\*Здесь должен был быть KIP ☹️\*



# Узкие места – Sender Thread



# Узкие места – Sender Thread

- Кардинальным образом влияет на Latency

# Узкие места – Sender Thread

- Кардинальным образом влияет на Latency
- Страдает от Lock Contention и «плохого» батчинга

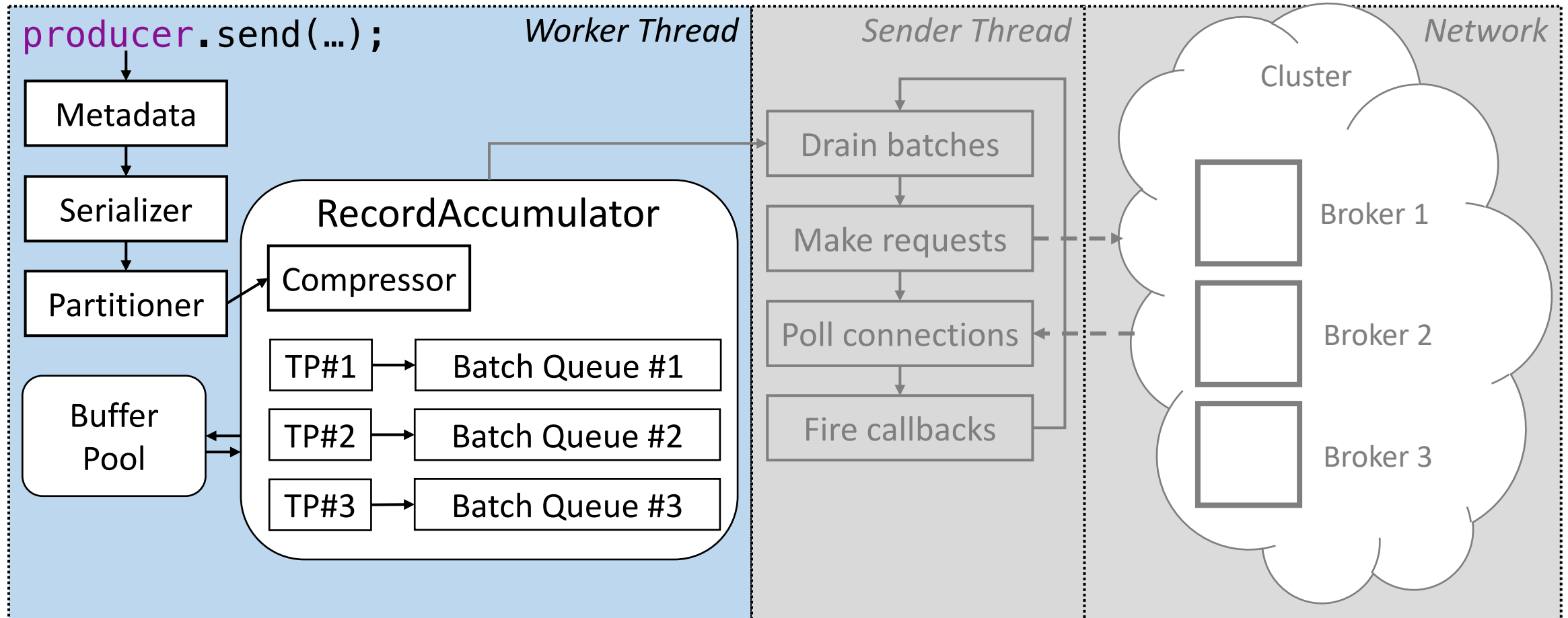
# Узкие места – Sender Thread

- Кардинальным образом влияет на Latency
- Страдает от Lock Contention и «плохого» батчинга
- Смотреть на **record-queue-time-avg** и **batch-size-avg**

# Узкие места – Sender Thread

- Кардинальным образом влияет на Latency
- Страдает от Lock Contention и «плохого» батчинга
- Смотреть на **record-queue-time-avg** и **batch-size-avg**
- Непрозрачная зависимость от Callback Latency

# Узкие места – Worker Thread



# Узкие места – Worker Thread

- Не влияет на Latency \*

# Узкие места – Worker Thread

- Не влияет на Latency \*
- Выбор **compression.type** влияет кардинально



# Узкие места – Worker Thread

- Не влияет на Latency \*
- Выбор **compression.type** влияет кардинально
- Увеличение пула потоков даёт заметный прирост Throughput

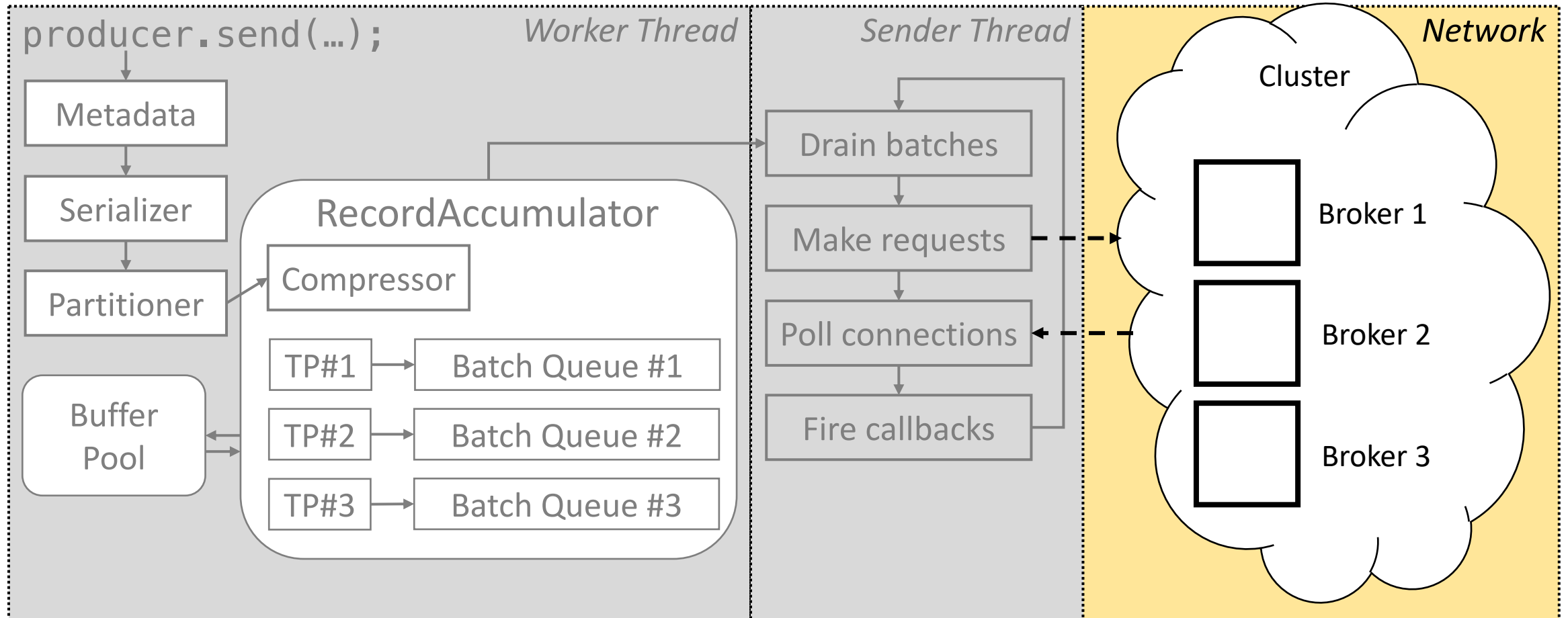
# Узкие места – Worker Thread

- Не влияет на Latency \*
- Выбор **compression.type** влияет кардинально
- Увеличение пула потоков даёт заметный прирост Throughput
- Страдает от Lock Contention сам и заставляет страдать Sender

# Узкие места – Worker Thread

- Не влияет на Latency \*
- Выбор **compression.type** влияет кардинально
- Увеличение пула потоков даёт заметный прирост Throughput
- Страдает от Lock Contention сам и заставляет страдать Sender
- «Правильный» Partitioner помогает Worker и Sender

# Узкие места – Broker и Network



# Узкие места – Broker и Network

Network

- Bandwidth
- RTT

# Узкие места – Broker и Network

Network

- **Bandwidth**

- RTT

1. Выбрать «лучший» `compression.type`

# Узкие места – Broker и Network

Network

- **Bandwidth**

- RTT

1. Выбрать «лучший» `compression.type`
2. 1 Gbps -> 10 Gbps

# Узкие места – Broker и Network

Network

- Bandwidth
- **RTT**

1. Улучшить батчинг



# Узкие места – Broker и Network

## Network

- Bandwidth

- **RTT**

1. Улучшить батчинг

2. Увеличить `send.buffer.bytes`

# Узкие места – Broker и Network

## Network

- Bandwidth
- **RTT**

1. Улучшить батчинг
2. Увеличить `send.buffer.bytes`
3. Увеличить `max.in.flight.requests.per.connection` \*

# Узкие места – Broker и Network

Broker

- Смотреть на **request-latency-avg**

# Узкие места – Broker и Network

Broker

- Смотреть на **request-latency-avg**

1. Неравномерное распределение партиций и/или лидерства

# Узкие места – Broker и Network

Broker

- Смотреть на **request-latency-avg**

1. Неравномерное распределение партиций и/или лидерства
2. Высокая утилизация дисков

# Узкие места – Broker и Network

Broker

- Смотреть на **request-latency-avg**

1. Неравномерное распределение партиций и/или лидерства
2. Высокая утилизация дисков
3. Медленный Fetch Follower

# Узкие места – Broker и Network

Broker

- Смотреть на **request-latency-avg**

1. Неравномерное распределение партиций и/или лидерства
2. Высокая утилизация дисков
3. Медленный Fetch Follower

```
num.replica.fetchers = 1
```

# Узкие места – Broker и Network

Вопросы для самопроверки



# Узкие места – Broker и Network

Вопросы для самопроверки

- Брокер пережимает данные? **compression.type = producer**

# Узкие места – Broker и Network

Вопросы для самопроверки

- Брокер пережимает данные? **compression.type = producer**
- Выбрано подходящее значения для **acks**?

# Узкие места – Broker и Network

Вопросы для самопроверки

- Брокер пережимает данные? **compression.type = producer**
- Выбрано подходящее значения для **acks**?
- На брокер приходит много запросов?  
Лучше батчинг – меньше нагрузка на брокера

# Узкие места – Broker и Network

Вопросы для самопроверки

- Брокер пережимает данные? **compression.type = producer**
- Выбрано подходящее значения для **acks**?
- На брокер приходит много запросов?  
    Лучше батчинг – меньше нагрузка на брокера
- Выбрано подходящее значение для **replication.factor**?

# Узкие места – Broker и Network

Вопросы для самопроверки

- Брокер пережимает данные? **compression.type = producer**
- Выбрано подходящее значения для **acks**?
- На брокер приходит много запросов?  
Лучше батчинг – меньше нагрузка на брокера
- Выбрано подходящее значение для **replication.factor**?
- Нужна ли гарантия порядка?  
**max.in.flight.requests.per.connection != 1**

# Настройки Producer одним слайдом

`acks`

`batch.size`

`linger.ms`

`partitioner.class`

`compression.type`

`max.in.flight.requests.  
per.connection`

`send.buffer.bytes`

`buffer.memory`

`max.request.size`

`retries`

`retry.backoff.ms`

`max.block.ms`

`request.timeout.ms`

`delivery.timeout.ms`

# Настройки Producer одним слайдом

`acks`

`batch.size`

`linger.ms`

`partitioner.class`

`compression.type`

`max.in.flight.requests.  
per.connection`

`send.buffer.bytes`

`buffer.memory`

`max.request.size`

**`retries`**


**`retry.backoff.ms`**

`max.block.ms`

`request.timeout.ms`

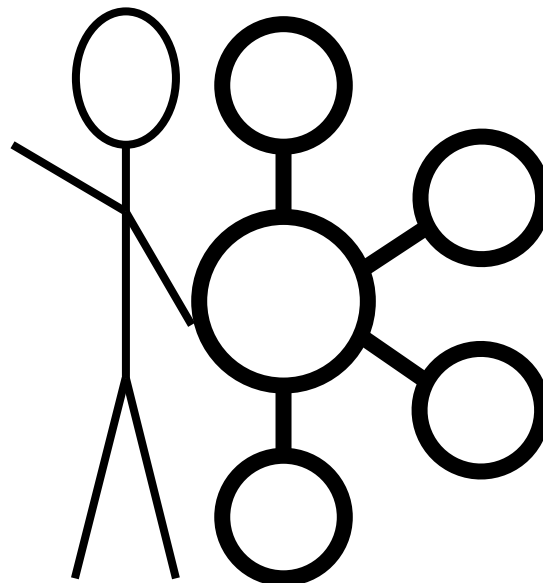
`delivery.timeout.ms`

 GregoryKoshelev

 K\_Gregory

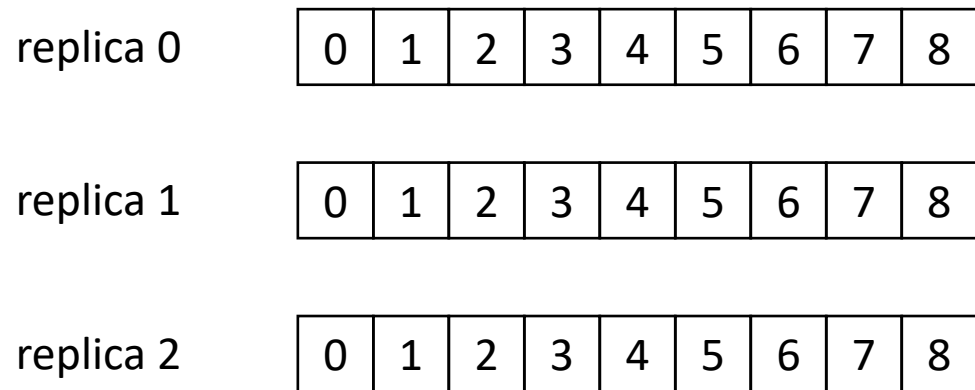
 gnkoshelev

[tech.kontur.ru](http://tech.kontur.ru)

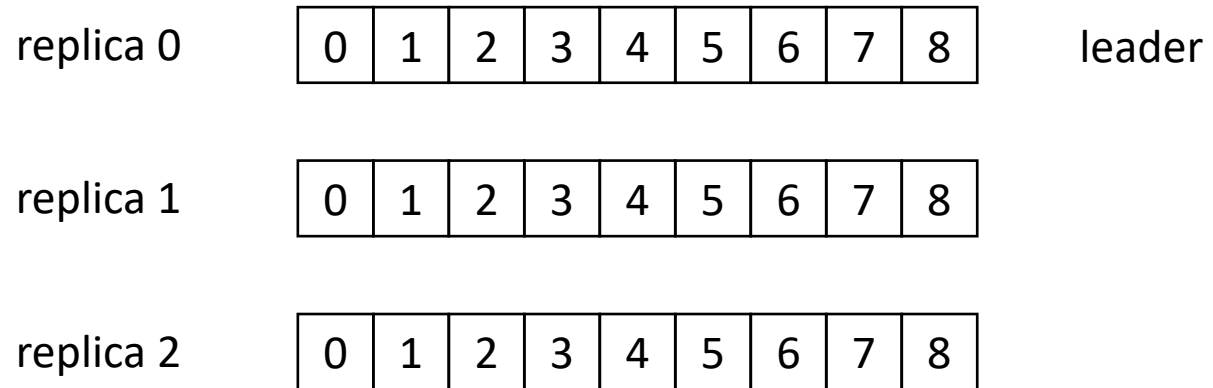




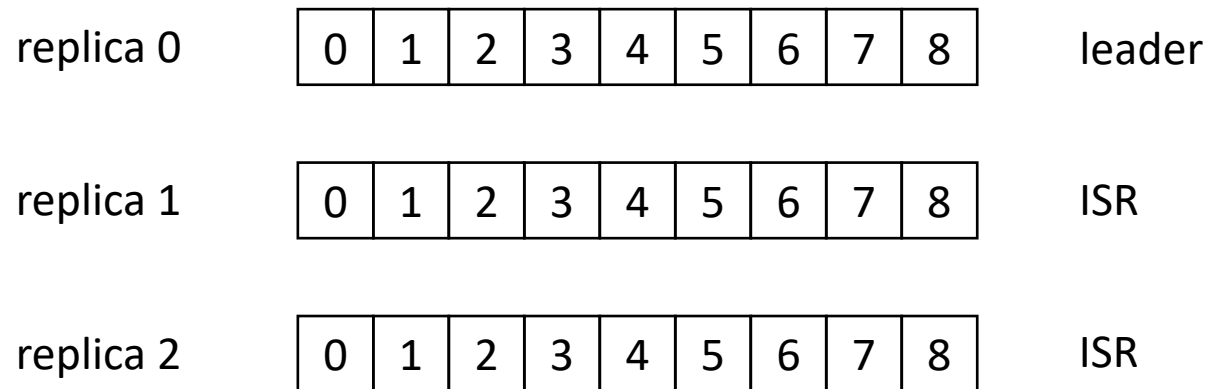
# Архитектура Kafka Producer



# Архитектура Kafka Producer

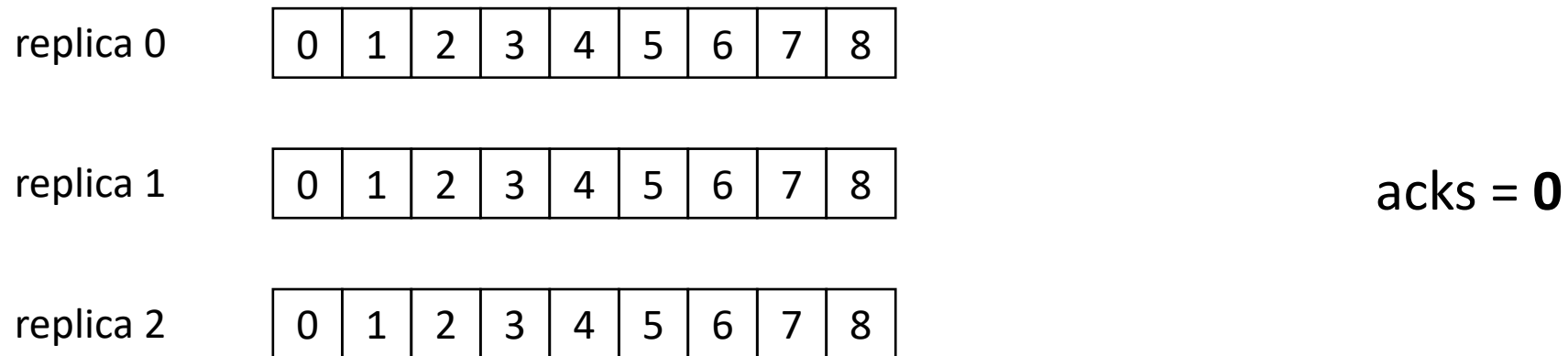


# Архитектура Kafka Producer



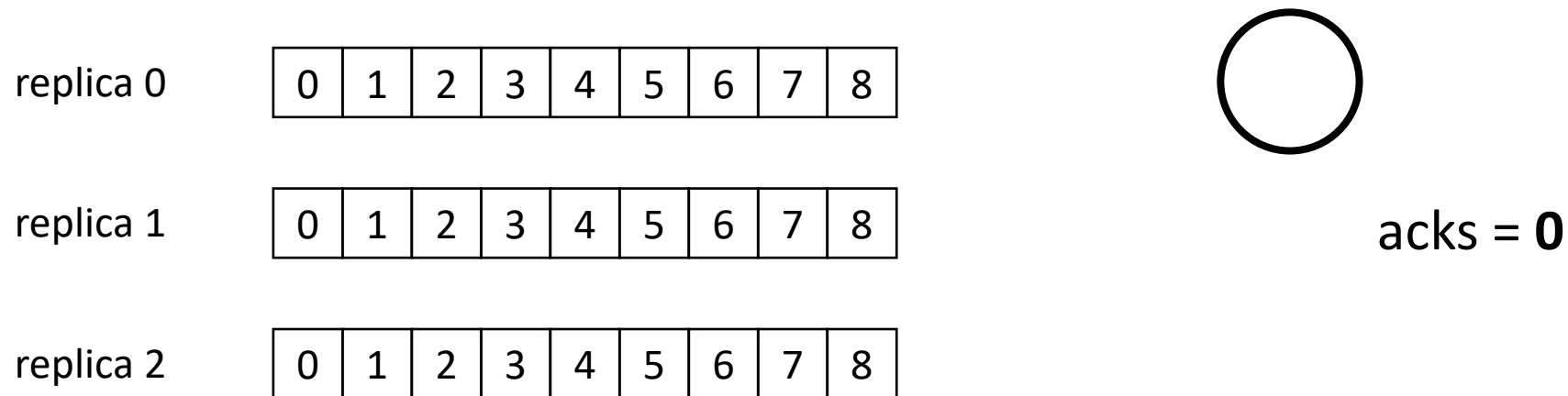
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



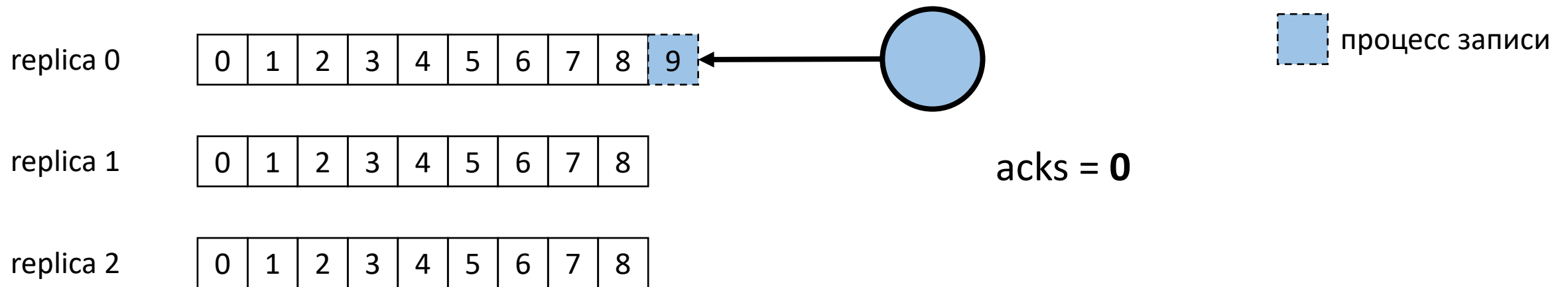
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



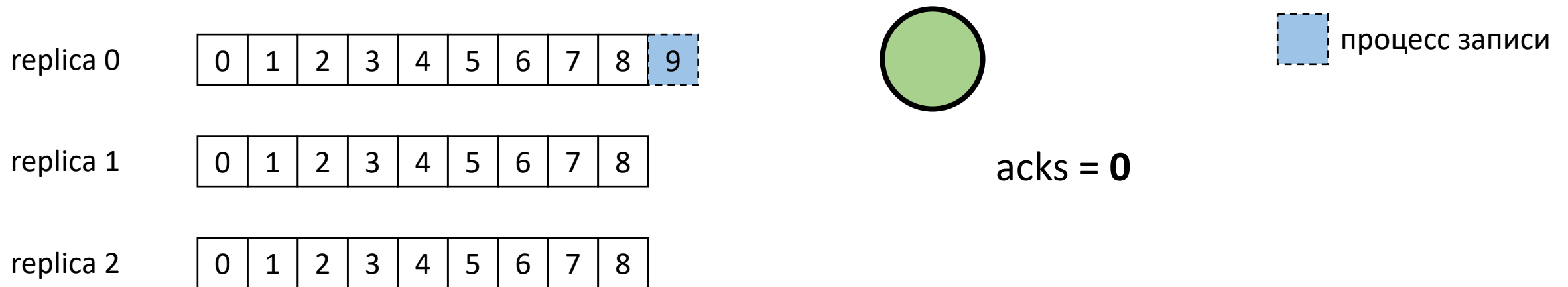
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



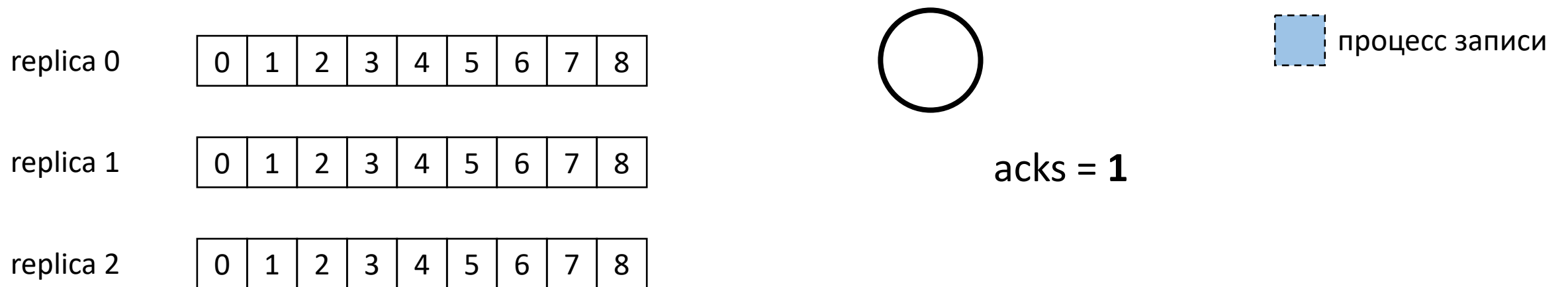
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



# Архитектура Kafka Producer

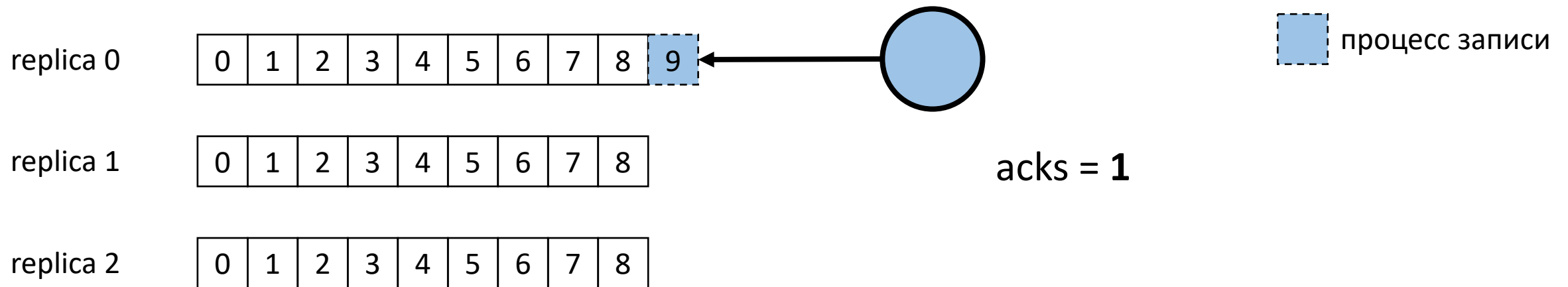
Acknowledgement (ack) – подтверждение записи





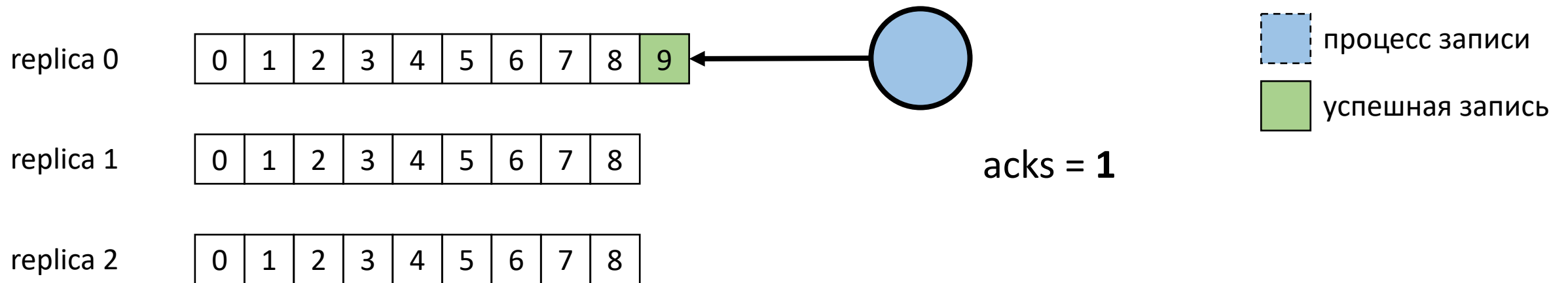
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



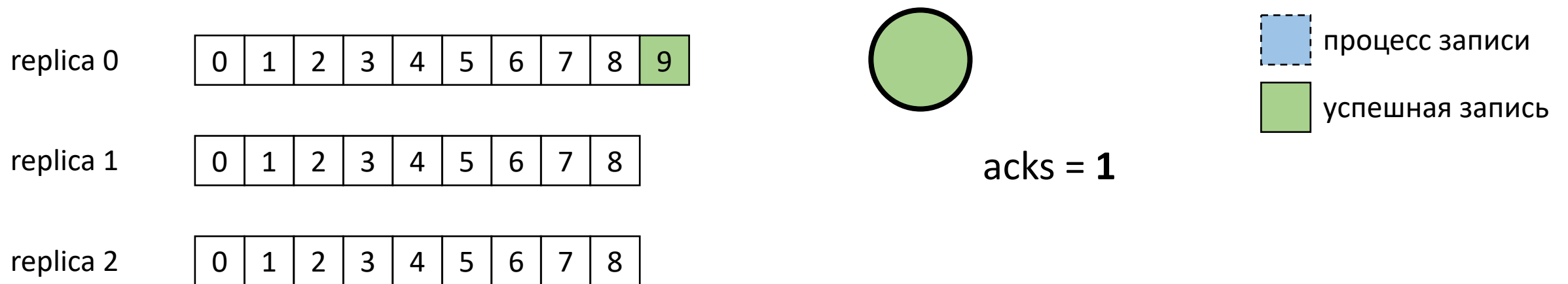
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



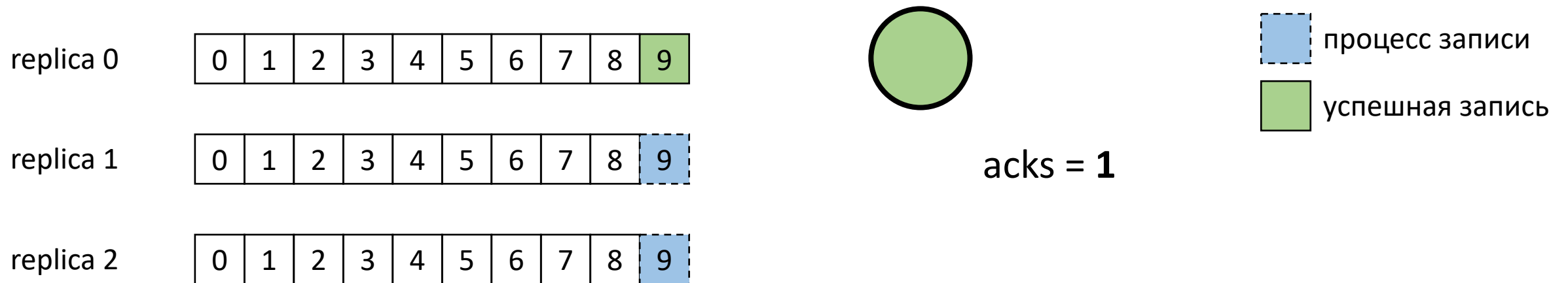
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



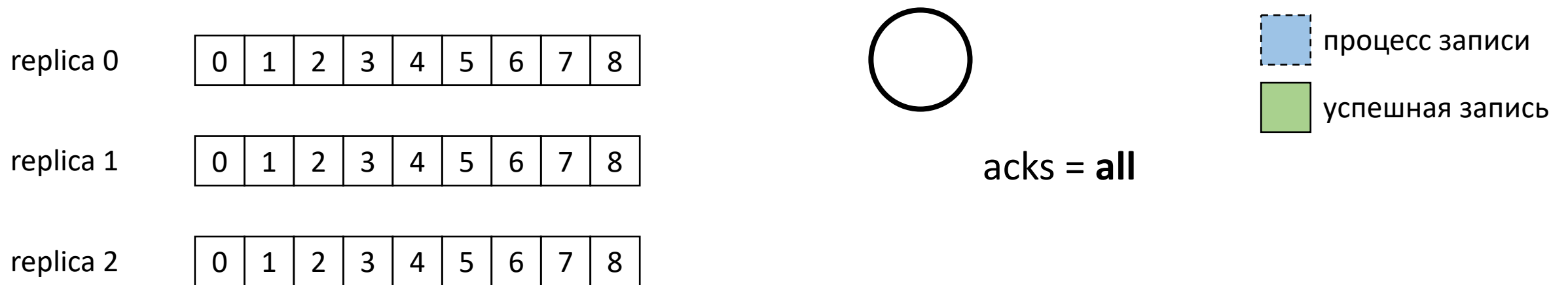
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



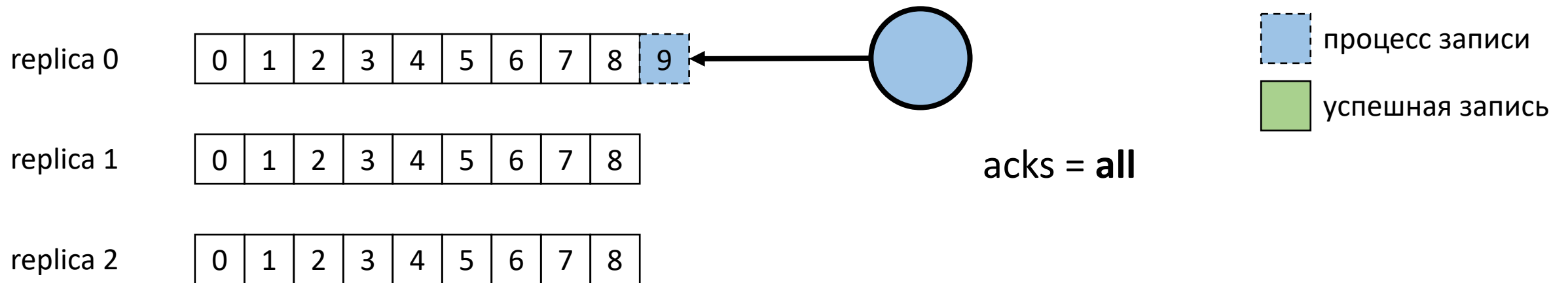
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



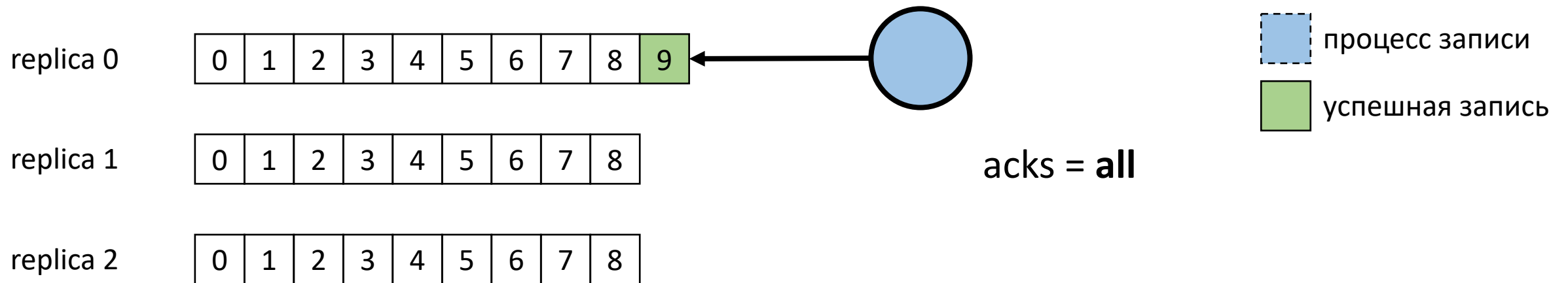
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



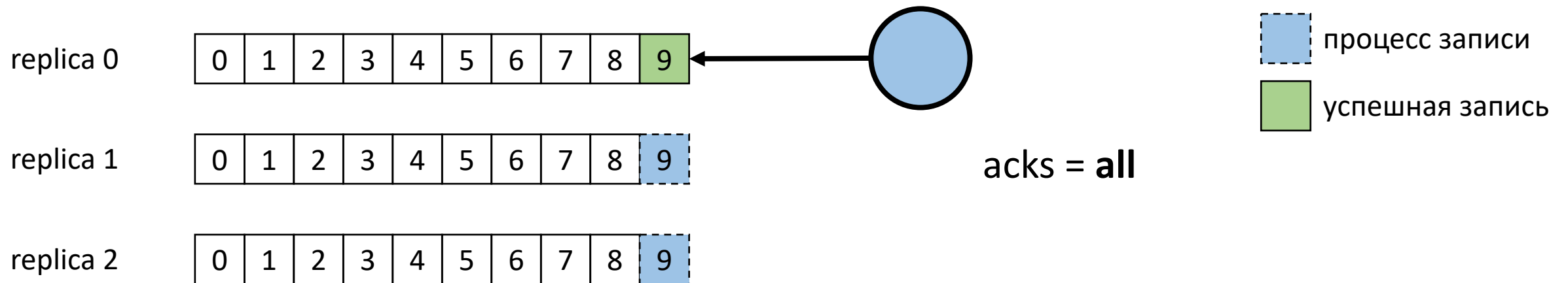
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



# Архитектура Kafka Producer

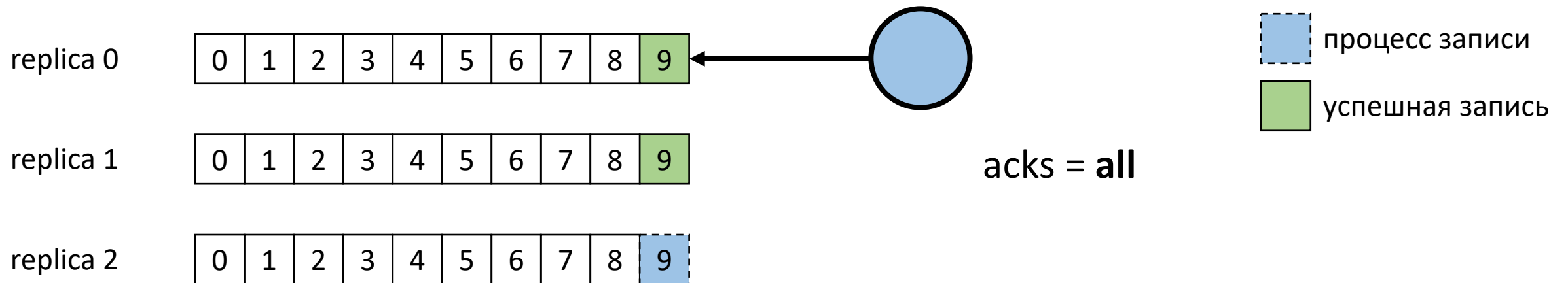
Acknowledgement (ack) – подтверждение записи





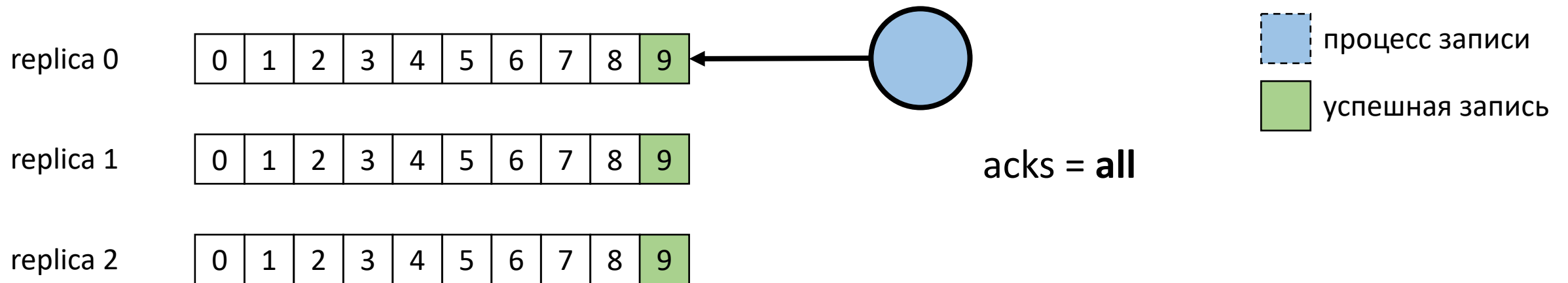
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



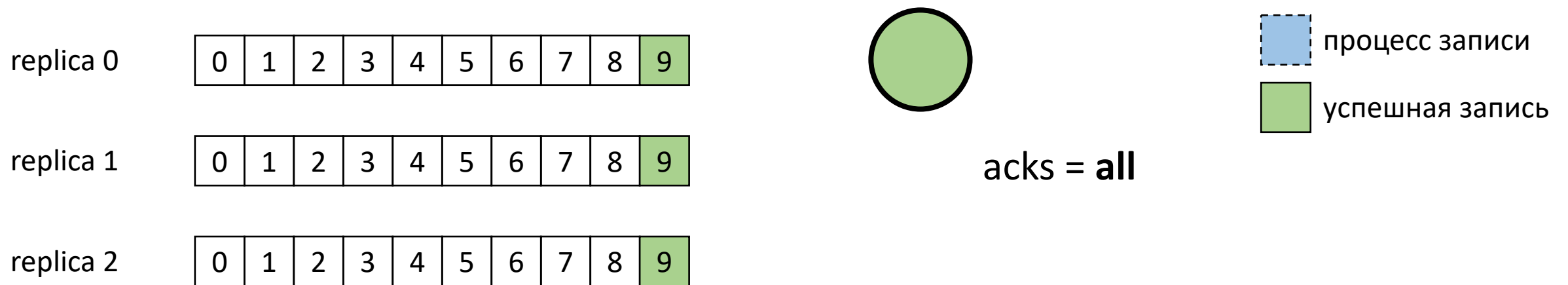
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



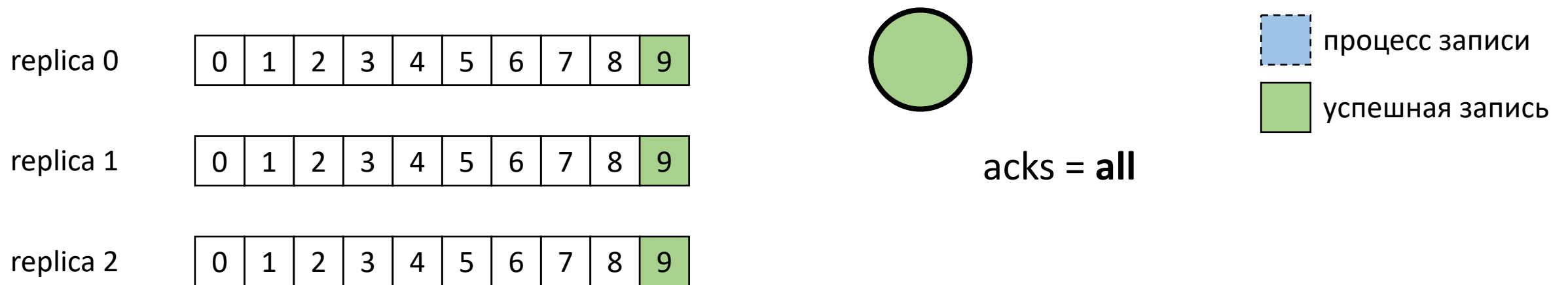
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



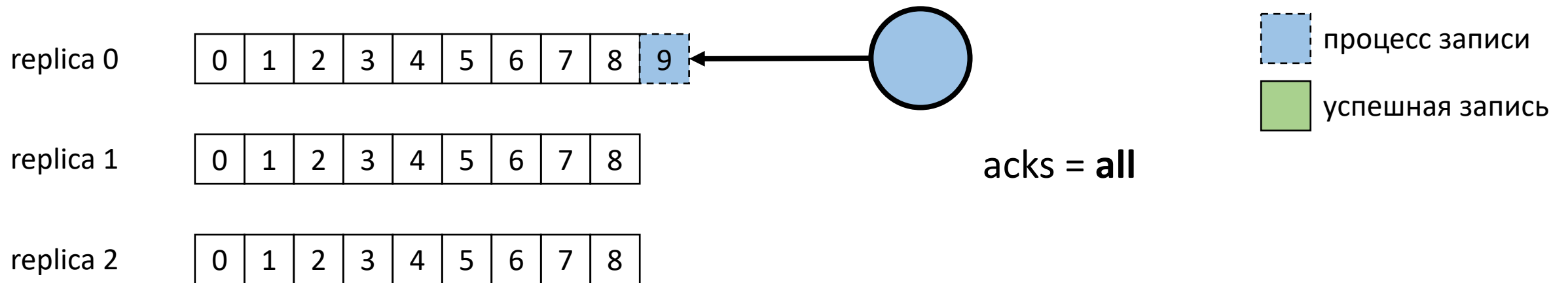
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



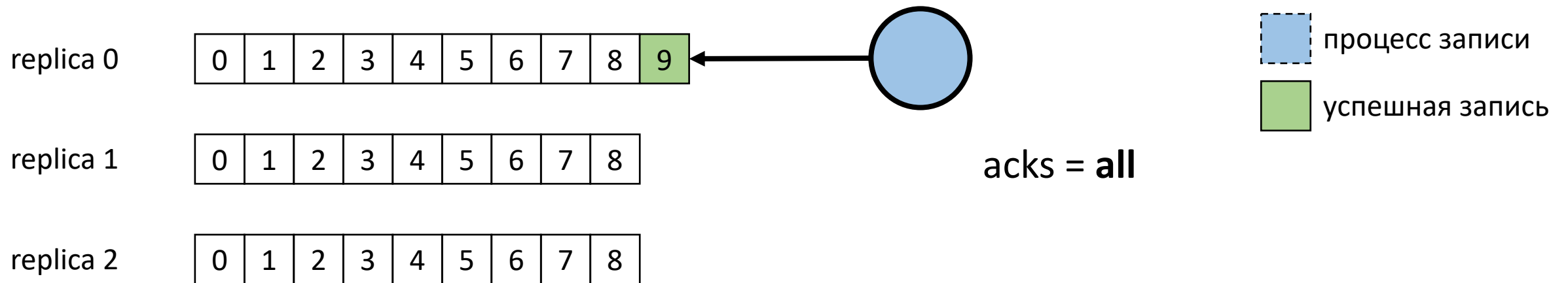
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



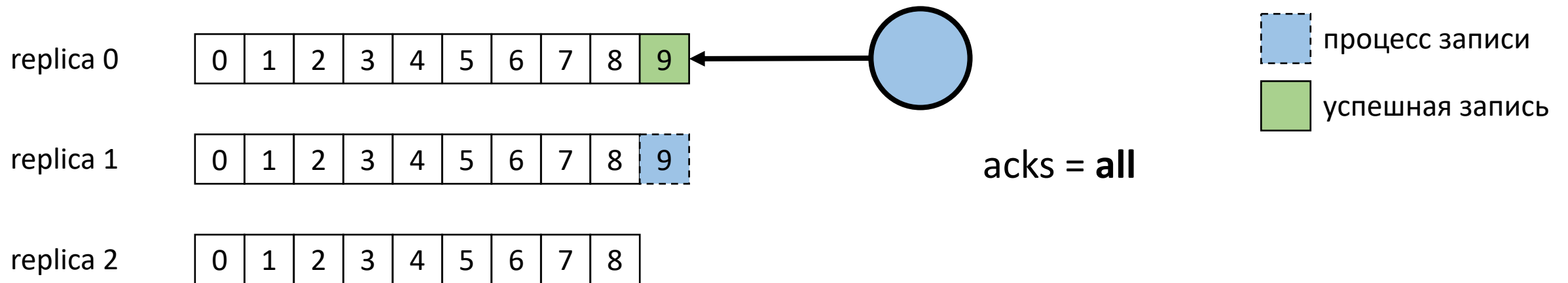
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



# Архитектура Kafka Producer

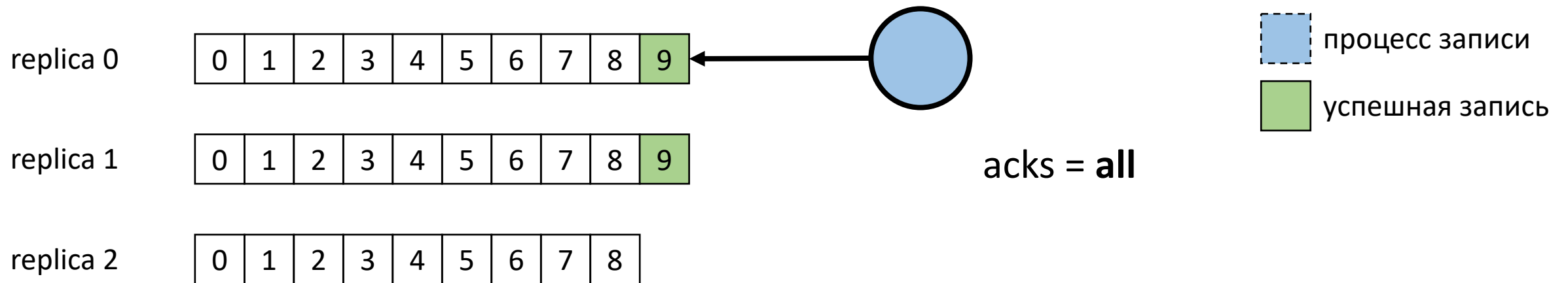
Acknowledgement (ack) – подтверждение записи





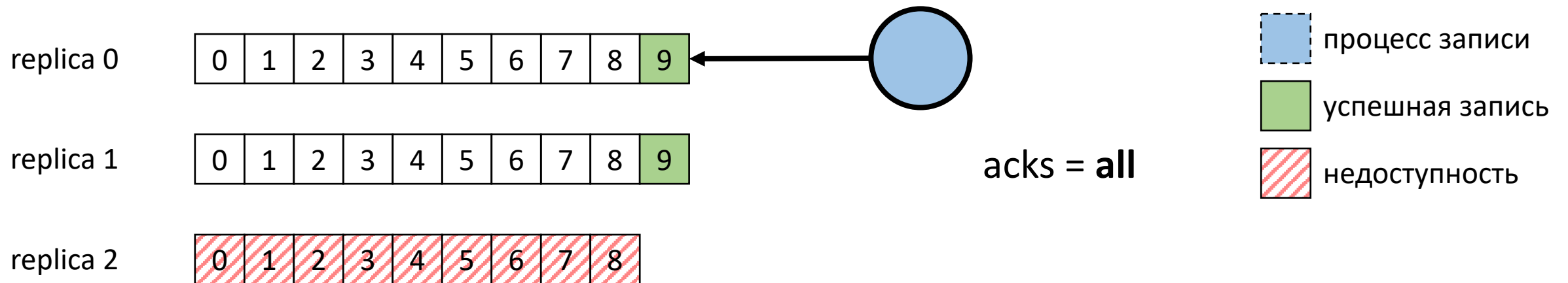
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



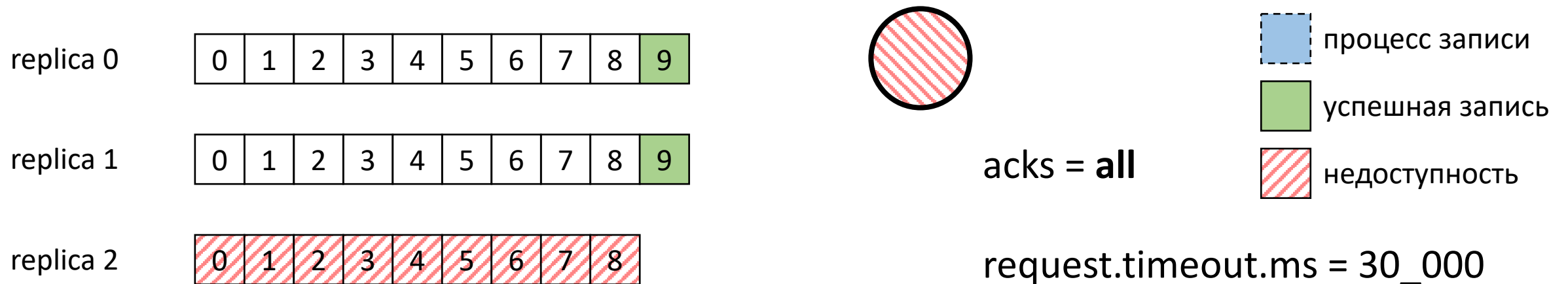
# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи



# Архитектура Kafka Producer

Acknowledgement (ack) – подтверждение записи

