thehoard.blog

How Kafka’s Storage Internals Work

Travis Jeffery

4-5 minutes

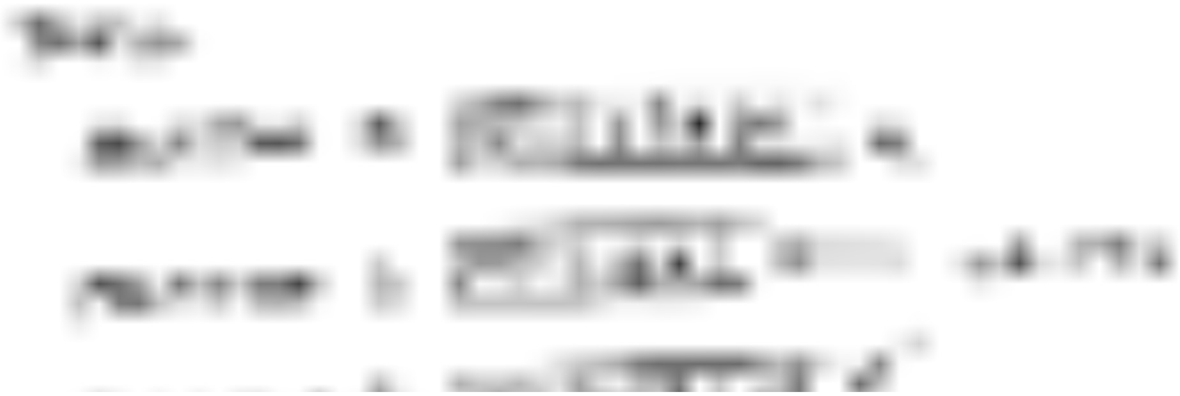
In this post I’m going to help you understand how Kafka stores its data.

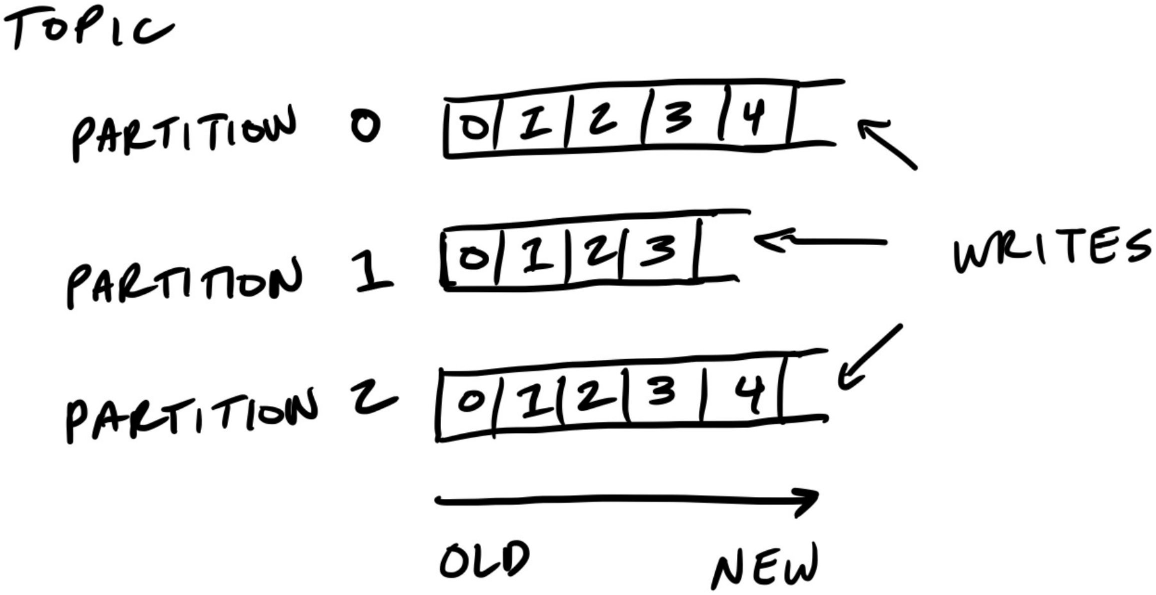
I’ve found understanding this useful when tuning Kafka’s performance and for context on what each broker configuration actually does. I was inspired by Kafka’s simplicity and used what I learned to start implementing Kafka in Golang.

So how does Kafka’s storage internals work?

Kafka’s storage unit is a partition

A partition is an ordered, immutable sequence of messages that are appended to. A partition cannot be split across multiple brokers or even multiple disks.





The retention policy governs how Kafka retains messages

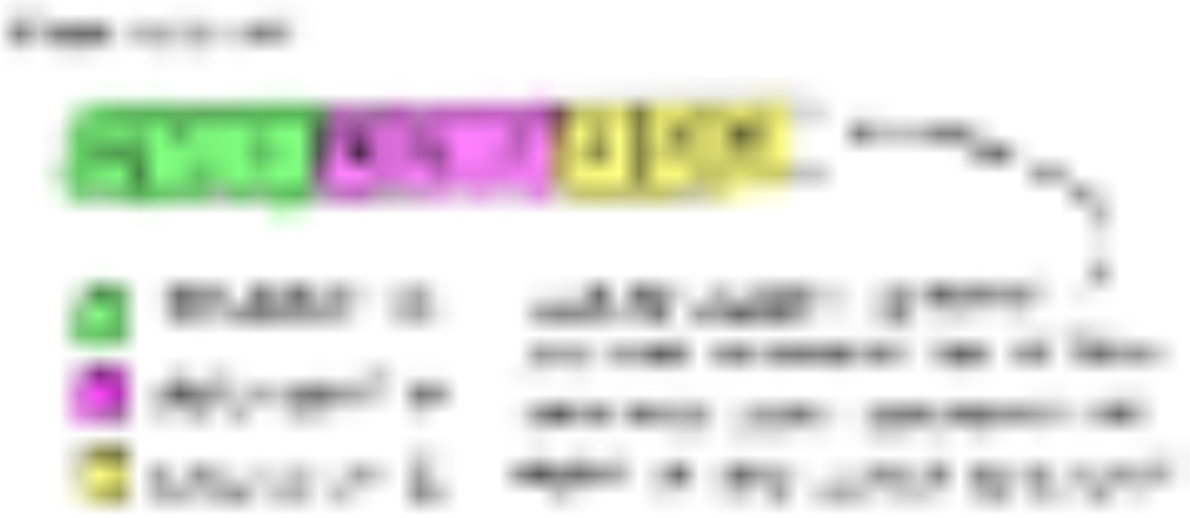
You specify how much data or how long data should be retained, after which Kafka purges messages in-order—regardless of whether the message has been consumed.

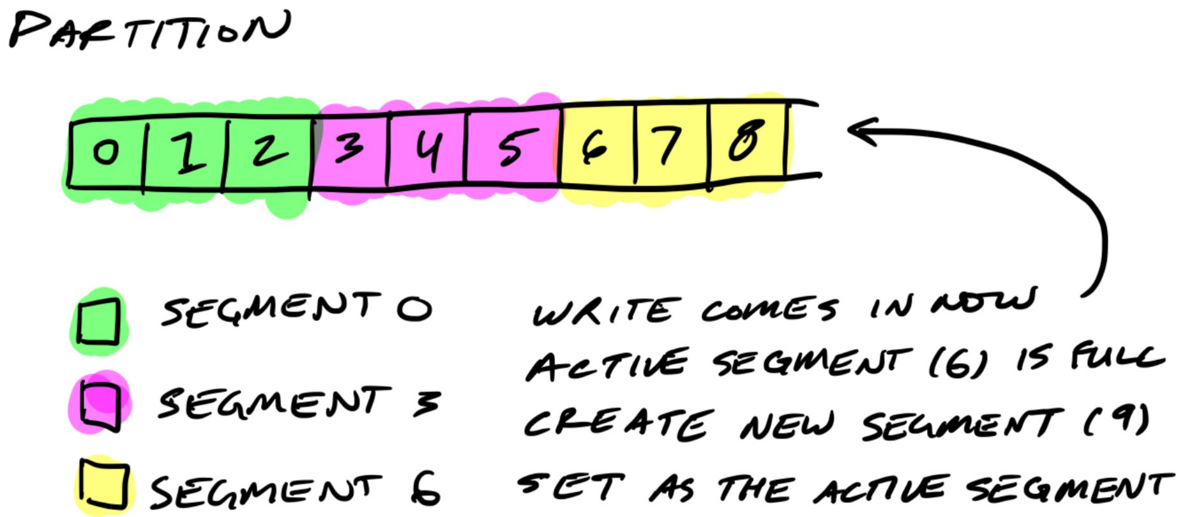
Partitions are split into segments

So Kafka needs to regularly find the messages on disk that need purged. With a single very long file of a partition’s messages, this operation is slow and error prone. To fix that (and other problems we’ll see), the partition is split into segments.

When Kafka writes to a partition, it writes to a segment — the active segment. If the segment’s size limit is reached, a new segment is opened and that becomes the new active segment.

Segments are named by their base offset. The base offset of a segment is an offset greater than offsets in previous segments and less than or equal to offsets in that segment.





On disk, a partition is a directory and each segment is an index file and a log file.

```
$ tree kafka | head -n 6
kafka
├── events-1
│ ├── 00000000003064504069.index
│ ├── 00000000003064504069.log
│ ├── 00000000003065011416.index
│ ├── 00000000003065011416.log
```

Segment logs are where messages are stored

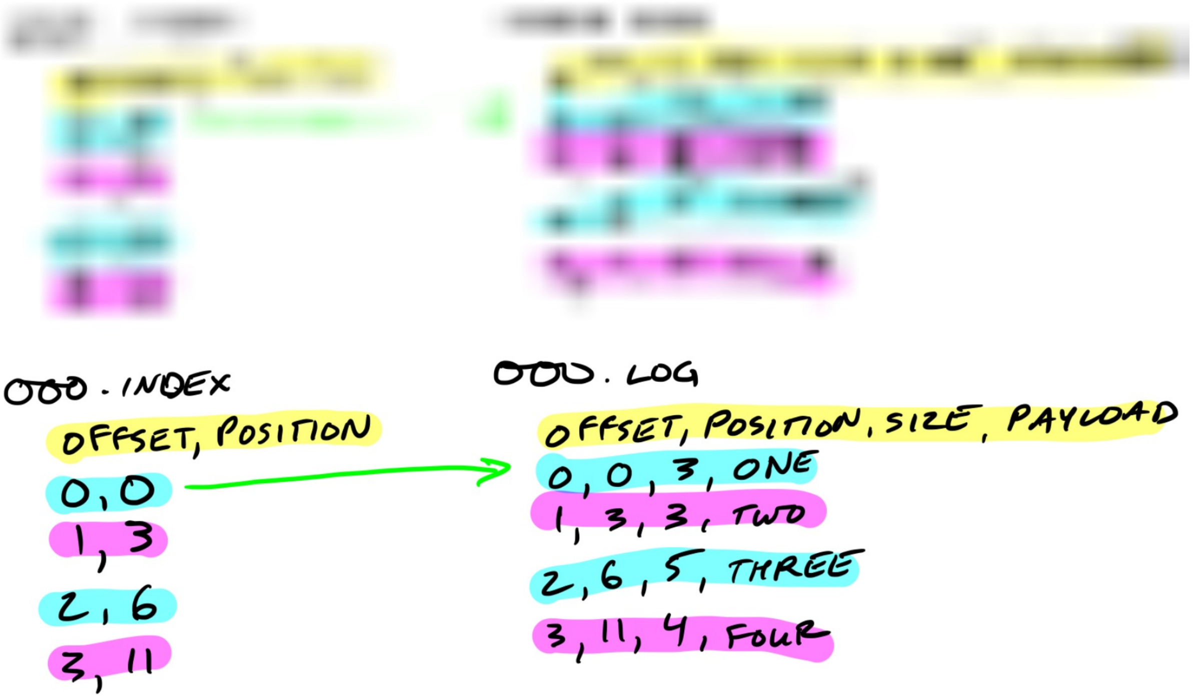
Each message is its value, offset, timestamp, key, message size, compression codec, checksum, and version of the message format.

The data format on disk is exactly the same as what the broker receives from the producer over the network and sends to its consumers. This allows Kafka to efficiently transfer data with zero copy.

```
$ bin/kafka-run-class.sh
kafka.tools.DumpLogSegments --deep-iteration
--print-data-log --files /data/kafka/events-
1/00000000003065011416.log | head -n 4
Dumping /data/kafka/appusers-
1/00000000003065011416.log
Starting offset: 3065011416
offset: 3065011416 position: 0 isvalid: true
payloadsize: 2820 magic: 1 compresscodec:
NoCompressionCodec crc: 811055132 payload:
{"name": "Travis", msg: "Hey, what's up?"}
offset: 3065011417 position: 1779 isvalid: true
payloadsize: 2244 magic: 1 compresscodec:
NoCompressionCodec crc: 151590202 payload:
{"name": "Wale", msg: "Starving."}
```

Segment indexes map message offsets to their position in the log

The segment index maps offsets to their message’s position in the segment log.



000.INDEX

OFFSET, POSITION
0, 0
1, 3
2, 6
3, 11

000.LOG

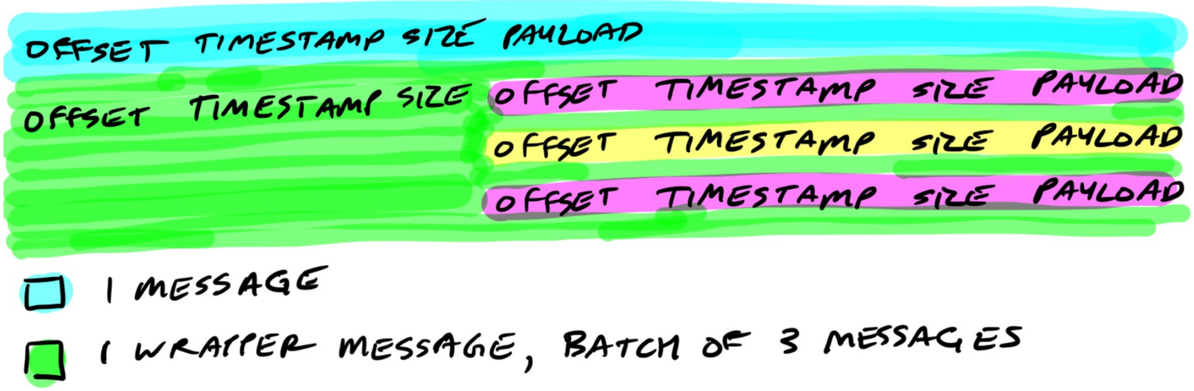
OFFSET, POSITION, SIZE, PAYLOAD
0, 0, 3, ONE
1, 3, 3, TWO
2, 6, 5, THREE
3, 11, 4, FOUR

The index file is memory mapped, and the offset look up uses binary search to find the nearest offset less than or equal to the target offset.

The index file is made up of 8 byte entries, 4 bytes to store the offset relative to the base offset and 4 bytes to store the position. The offset is relative to the base offset so that only 4 bytes is needed to store the offset. For example: let’s say the base offset is 10000000000000000000, rather than having to store subsequent offsets 10000000000000000001 and 10000000000000000002 they are just 1 and 2.

Kafka wraps compressed messages together

Producers sending compressed messages will compress the batch together and send it as the payload of a wrapped message. And as before, the data on disk is exactly the same as what the broker receives from the producer over the network and sends to its consumers.


OFFSET TIMESTAMP SIZE PAYLOAD
OFFSET TIMESTAMP SIZE
OFFSET TIMESTAMP SIZE PAYLOAD
OFFSET TIMESTAMP SIZE PAYLOAD
OFFSET TIMESTAMP SIZE PAYLOAD
1 MESSAGE
1 WRAPPER MESSAGE, BATCH OF 3 MESSAGES

Let’s Review

Now you know how Kafka storage internals work:

- Partitions are Kafka’s storage unit
- Partitions are split into segments
- Segments are two files: its log and index
- Indexes map each offset to their message’s position in the log, they’re used to look up messages
- Indexes store offsets relative to its segment’s base offset
- Compressed message batches are wrapped together as the payload of a wrapper message
- The data stored on disk is the same as what the broker receives from the producer over the network and sends to its consumers