Compression

This feature introduces the end-to-end block compression feature in Kafka. If enabled, data will be compressed by the producer, written in compressed format on the server and decompressed by the consumer. Compression will improve the consumer throughput for some decompression cost. This is especially useful when mirroring data across data centers.

Design

A set of messages can be compressed and represented as one compressed message. In that sense, a message is recursive by definition. A ByteBufferMessageSet could consist of both uncompressed as well as compressed data. Due to that, we need some way of identifying compressed messages from uncompressed ones. To identify a compressed message, we introduce a compression-attributes byte in the message header. This addition of a byte in the message header indicates change in the network byte format as well as the storage byte format. This new format is versioned as magic byte value of 1.

The message header format for magic byte=1, now looks like -

1 byte magic	1 byte compression-attributes	4 byte CRC32 of the payload

The lowest 2 bits in the attributes byte will select the compression codec used for the compressing data. A compression codec value of 0 indicates an uncompressed message.

Offset management on the consumer

The data received by a consumer for a topic might contain both compressed as well as uncompressed messages. The consumer iterator transparently decompresses compressed data and only returns an uncompressed message. The offset maintenance in the consumer gets a little tricky. In the zookeeper consumer, the consumed offset is updated each time a message is returned. This consumed offset should be a valid fetch offset for correct failure recovery. Since data is stored in compressed format on the broker, valid fetch offsets are the compressed message boundaries. Hence, for compressed data, the consumed offset will be advanced one compressed message at a time. This has the side effect of possible duplicates in the event of a consumer failure. For uncompressed data, consumed offset will be advanced one message at a time.

Backwards compatibility

A version 0.7 broker and consumer will be able to understand messages of magic byte values 0 and 1. So the brokers and consumers are backwards compatible.

Configuration changes

There are 2 new config parameters on the producer side -

Config parameter	Description	Default
compression.codec	Controls the compression codec to be used by the producer. (0: No compression, 1: GZIP compression, 2: Snappy compression, 3: LZ4 compression)	0
compressed.topics	comma separated list of topics for which compression should be enabled. This doesn't mean anything when compression.codec = 0	empty

	compression.topics=empty	compression.topics="topicA,topicB"
compression.codec=0	All topics are uncompressed since compression is disabled	All topics are uncompressed since compression is disabled
compression.codec=1	All topics are compressed	Only the topics topicA and topicB are compressed

Compression codecs supported

Currently, only GZIP, Snappy and LZ4 compression codecs are supported.