KIP-57 - Interoperable LZ4 Framing

Status

Current state: *Accepted*

Discussion thread: here

JIRA: here

Github PR: PR 1212

Released: 0.10.0.0

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Kafka's initial LZ4 compression implementation is not interoperable. It does not follow the standard LZ4 framing specification (see https://cyan4973.github.io/lz4/lz4_Frame_format.html). This makes it difficult for third-party clients to support LZ4 compression using off-the-shelf libraries. This KIP proposes to fix kafka's LZ4 handling so that it is conformant with the LZ4F specification and enable clients to interoperate with respect to LZ4-compressed messages.

Specifically, KAFKA-1493 attempted to implement the LZ4F interoperable framing specification. There's a bug, however, that causes the frame checksum to be incorrectly calculated. Fixing this single byte (refered to as HC) is the goal of this KIP.

Public Interfaces

Changes to public interfaces:

None. Interface changes are for classes not currently marked as public in javadoc.

Proposed Changes

Old 0.8.2/0.9 clients (current behavior):

- produce messages w/ broken HC checksum only
- consume messages w/ incorrect HC checksum only

Old 0.8.2/0.9 brokers (current behavior):

- requires that all lz4 compressed messages must have incorrect HC checksum
- error code on incorrect HC checksum is -1 (UnknownError)

New 0.10 clients (proposed behavior)

- produce and consume v1 messages w/ correct LZ4F checksum

New 0.10 broker (proposed behavior):

- Return v0 (KIP-31) messages w/ old "broken" checksum in FetchResponse
  - v0 on-disk stored with incorrect checksum, returned directly
  - use KIP-31 conversion to "break" checksum when stored as v1 on-disk with correct checksum
- Return v1 (KIP-31) messages w/ correct checksum in FetchResponse
  - v1 on-disk format stored with correct checksum
  - use KIP-31 conversion to "fix" checksum when stored as v0 on-disk with incorrect checksum
- reject v1 produced messages that do not have correct checksum
- disable checksum validation for v0 produced messages
- all LZ4 errors return code 2 in ProduceResponse (Invalid/Corrupt Message)

KafkaLZ4* code:

- fix checksum calculation for both compress and decompress
- add option to compression class to allow writing incorrect checksum for compatibility
- add option to decompression class to allow ignoring incorrect checksum for compatibility
- do not reject messages that have optional lz4 header flags set: ContentSize or ContentChecksum. This is to enable interoperability with off-the-shelf lz4 libraries that may set them. The only flag left unsupported is dependent-block compression (LZ4 Stream API), which our implementation does not currently support.

Compatibility, Deprecation, and Migration Plan

- Compatibility with old clients is maintained by switching the LZ4 framing checksum behavior with the v0 / v1 message format. This allows old clients to continue producing and consuming LZ4 messages in the old format, and enables new clients to produce and consume messages in the new format. It also leverages the KIP-31 v1<->v0 "conversion" process to reencode LZ4 messages as required to either fix (v0->v1) or break (v1->v0) the checksum.
- The one use case that may need special attention is trunk users who use LZ4 compressed messages AND have already upgraded to v1 messages. Are there any such users? For these users, upgrading brokers would cause their prior v1 producers to immediately fail because broker will reject v1 messages with broken checksum. Upgrading producers will fix and enable production to continue. If any such users exist, it might be worth providing a one-time patch to apply to trunk that disables the broker error on v1 messages with broken HC. Alternately we could allow this to be configured, but I think we should be skeptical of this as it will likely create more confusion for the vast majority of users who are not in this edge-case. There have been no reports of this usage on the kafka mailing lists during KIP discussion.
- Note that because the broker would no longer validate HC checksums on v0 messages, it will be possible for (non-java) clients to produce LZ4 messages in v0 format using a correct checksum. The broker will alter the checksum so that it appears "broken" for compatibility with older clients during KIP-31 conversion. Clients wishing to consume v0 format LZ4-messages must either ignore the HC checksum locally, or implement the broken HC checksum logic (also apply checksum to 4-byte magic header)

Rejected Alternatives

Alternative #1: Create a new compression type, "LZ4F" . Rejected because this is really just a bugfix, not a new compression type. The number of compression types is limited by the number of bits available in message attribute byte. We currently use 2 bits to cover the 4 compression types (None, Gzip, Snappy, LZ4). Adding a second type for a "fixed" LZ4 would require pulling a 3rd bit from attributes bytes. Further, explaining to users the difference between LZ4 and LZ4F compression types is likely to be difficult.