KIP-390: Support Compression Level

- Status
- Motivation
- Public Interfaces
- Proposed Changes
- Benchmark
    - Produce Test
        - Producer
        - Data
        - Environment
        - Broker/Topic
    - Result
    - Linear Write Test
    - Result
- Compatibility, Deprecation, and Migration Plan
- Further works
    - Compression buffer size option
    - Long window size with Zstandard
- Rejected Alternatives

Status

Current state: Accepted

Discussion thread: here

JIRA: KAFKA-7632

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

Motivation

Basically, CPU (running time) and I/O (compressed size) are trade-offs in compression. Since the best is use case dependent, lots of compression libraries provide a way to control the compression level with a reasonable default level, which results in a good performance in general. However, Kafka does not provide a way to configure the compression level - it uses the default level only.

This proposal suggests adding the compression level option to the producer, broker, and topic config. Running tests with a real-world dataset (see below), I found that this option improves the producer's message/second rate up to 156%.

Public Interfaces

This feature introduces 3 new options, 'compression.gzip.level','compression.lz4.level' and 'compression.snappy.level' to the producer, topic, and broker configuration. Snappy is excluded since it does not support any compression level. The type of these options is an integer.

For example

```
compression.type=gzip
compression.gzip.level=4                                # NEW: Compression level to be used.
```

The table below shows the valid range of level per compression.type. If the level is not explicitly set by the user, the default value from the compression library will be used like it is done today. The valid range and default value of the compression level are entirely up to the compression library, so they may be changed in the future.

Compression Codec	Valid Range
gzip	1 (Deflater.BEST_SPEED) ~ 9 (Deflater.BEST_COMPRESSION)
lz4	1 ~ 17
zstd	-131072 ~ 22

Proposed Changes

This option impacts the following processes:

- Producer compresses the user-given messages.
- Broker recompresses the user-given messages with specified compression.type per broker or topic.
- Broker recompresses the messages in the log cleanup process.

Compressing the records with the given compression type and level works like the following:

- If 'compression.type' is none or snappy, 'compression.<codec>.level' is ignored.
- If 'compression.<codec>.level' is not in the valid range, it raises an error.
- If 'compression.<codec>.level' is in the valid range, the producer compresses the records with the given level.
- If 'compression.<codec>.level' is not set, it falls back to the default level.

Benchmark

Produce Test

To benchmark how compression level affects the producer performance, I ran a small benchmark with a real-world dataset like below:

Producer

With the feature implemented on top of the latest trunk (commit ccec9b0), I ran kafka-producer-perf-test.sh on GraalVM Java 8 v21.1.0 with the following parameters:

- Number of records: 100,000
- batch.size: 1048576 (1mb)
- linger.ms: 100

Data

A random sample of 4096 real-world records from this dataset, which consists of 129218 json files with an average size of 55.25kb.

Environment

MS Azure Kubernetes Cluster (Seoul Region), consists of 16 nodes of Standard_DS2_v2 (2vCPU, 7GB RAM, Expected network bandwidth of 1500 Mbps.)

Broker/Topic

Apache Kafka 2.7.0, GraalVM Java 8 (21.1.0), replicaton factor = 3.

Result

codec	level	produced message / sec	latency (ms)	size (bytes)	description
none		2,739.50	205.34	5,659,454,754	
gzip	1	1,122.96	1,230.22	1,787,505,238	min. level
gzip	6	717.71	2,041.24	1,644,280,629	default level
gzip	9	608.54	2,413.66	1,643,517,758	max. level
lz4	1	1,694.69	603.46	2,211,346,795	min. level
lz4	9	1,199.93	878.85	2,184,022,257	default level
lz4	17	495.34	2,110.55	2,178,643,665	max. level
zstd	-5	7,653.45	156.88	1,997,500,892	experimental level
zstd	1	6,317.52	68.55	1,521,783,958	
zstd	3	4,760.54	286.79	1,494,620,615	default level
zstd	12	988.95	863.89	1,458,150,768	
zstd	18	85.20	2,017.92	1,492,015,424	

It shows the following:

- Codec is the main factor that differentiates the compressed size. However, The compression level makes little impact on it. The maximum improvement is is gzip/1 vs. gzip/9 (8%), and the minimum is lz4/1 vs. lz/17 (1.5%).
- Excepting zstd/-5, when the compression level gets lower, messages/sec increase but latency decreases. Especially, **compressing with zstd/1 produces 32.7% more messages per second than zstd/3 (current default), and gzip/1 produces 56.4% than gzip/6 (current default).**
- **For every compression codec, compression with minimum level (i.e., speed first strategy) resulted in the best messages/second rate.**

Linear Write Test

To benchmark how compression level affects the linear write performance, I ran a small benchmark with a real-world dataset like below:

```
INCLUDE_TEST_JARS=true bin/kafka-run-class.sh kafka.TestLinearWriteSpeed --bytes 8192 --size 8192 --message-size 4096 --files 1 --compression {compression-codec} --level {compression-level} --log
```

Result

codec	level	write speed (mb/sec)	description
none		19678.841	
gzip	1	22007.042	min. level
gzip	6	18425.707	default level
gzip	9	19148.284	max. level
lz4	1	22776.967	min. level
lz4	9	20613.456	default level
lz4	17	19879.134	max. level
zstd	-5	19531.25	experimental level
zstd	1	22910.557	
zstd	3	19531.25	default level
zstd	12	17477.628	
zstd	18	21229.619	

The result was almost similar. In general, the minimum compression level (=1) showed the best write speed (except zstd/-5).

Compatibility, Deprecation, and Migration Plan

Since this update follows the default compression level and current buffer size if they are not set, there is no backward compatibility problem.

Further works

Alongside the compression level, I am trying additional configuration options, like the following:

Compression buffer size option

At the initial stage, the compression buffer size option was also under consideration. However, during the benchmark, I could not find its positive impacts on produce speed or compressed size. I am still investigating whether it can improve the disk write speed, e.g., Broker-size recompression or compaction.

Long window size with Zstandard

With 1.3.2, Zstandard introduced compression/decompression with long window size. This option can improve the compression/decompression speed for some levels.

Rejected Alternatives

- Introduce a single configuration: compression.level. This was the original proposal that was voted. However while reviewing the PR we noticed that it was hard to use as each codec has its own range of compression levels. So for example setting compression.level to 20 would be valid with zstd but an error with gzip. Also a follow up KIP, KIP-780, aims at introducing per codec configurations using the compression.<codec>.<option> syntax. For these reason, it seems preferable to use configurations with the 'compression.<codec>.level' format for consistency.