KIP-640: Add log compression analysis tool

Status

Current state: Under Discussion

Discussion thread: here

JIRA: ⚠ Unable to render Jira issues macro, execution error.

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

Motivation

Compression is often used in Kafka to trade off extra CPU usage in Kafka clients for reduced storage and network resources on Kafka brokers. Compression is most commonly configured to be done by producers, though compression can also be configured to be performed by the brokers for situations where producers do not have spare CPU cycles. Regardless of the configuration used, the compression algorithm chosen will vary depending upon the needs of each use case.

To determine which compression algorithm to use, it is often helpful to be able to quantify the savings in storage, ingress bandwidth (if any), replication bandwidth, and egress bandwidth, all of which are a function of how much the compression algorithm reduces the overall size of the messages. Because the performance characteristics of each compression algorithm are highly dependent on the data being compressed, measuring the reduction in data size typically requires the user to produce data into Kafka using each compression algorithm and measure the resulting bandwidth utilization and log size for each use case. This process is time consuming and if the user is not careful, can easily provide vague or misleading results.

Public Interfaces

A new command line tool called `kafka-compression-analyzer.sh` that measures what the size of a log segment would be after compressing it using each of the compression types supported by Kafka. It is a read-only tool and does not modify the log segment being analyzed. This tool will will accept several command line parameters:

Parameter	Required	Description
--logs	Yes	The comma-separated list of log files to be analyzed.
--verbose	No	If set, display verbose batch information.

Output

The tool will print results to standard out. The tool reports information about the batches in the log segment (as more batching often helps improve the effectiveness of compression), the breakdown of compression types found in the log segment, and the results of applying each compression type. A sample output:

Sample Output

```
Analyzing /kafka/test-topic-0/00000000000525233956.log
Original log size: 536793767 bytes
Uncompressed log size: 536793767 bytes
Original compression ratio: 1.00
Original space savings: 0.00%


Batch stats:
  16593/20220 batches contain >1 message
  Avg number of messages per batch: 3.68
  Avg batch size (original): 5180 bytes
  Avg batch size (uncompressed): 5180 bytes

Number of input batches by compression type:
  none: 20220

COMPRESSION-TYPE  COMPRESSED-SIZE  SPACE-SAVINGS  COMPRESSION-RATIO  AVG-RATIO/BATCH  TOTAL-TIME        SPEED
gzip                    118159324         22.01%              4.543            1.795     13875ms   36.90 MB/s
snappy                  160597012         29.92%              3.342            1.549      2678ms  191.16 MB/s
lz4                     161711232         30.13%              3.319            1.576      2616ms  195.69 MB/s
zstd                    112737048         21.00%              4.761            1.775      5103ms  100.32 MB/s
```

Sample Output 2

```
Analyzing /kafka/test-topic-1/00000000000000000000.log
Original log size: 14510269 bytes
Uncompressed log size: 16080153 bytes
Original compression ratio: 1.11
Original space savings: 9.76%

Batch stats:
  6/2875 batches contain >1 message
  Avg messages/batch: 1.01
  Avg batch size (original): 1255 bytes
  Avg batch size (uncompressed): 3125 bytes

Number of input batches by compression type:
  none: 1784
  gzip: 525
  snappy: 275
  lz4: 291

COMPRESSION-TYPE  COMPRESSED-SIZE  SPACE-SAVINGS  TOTAL-RATIO  AVG-RATIO/BATCH  TOTAL-TIME        SPEED
gzip                       422829         97.37%        38.03            21.43       168ms   91.28 MB/s
snappy                    1103867         93.14%        14.57            10.30        45ms  340.78 MB/s
lz4                        423965         97.36%        37.93            21.46       195ms   78.64 MB/s
zstd                       352861         97.81%        45.57            25.46       251ms   61.10 MB/s
```

Breakdown of outputs:

Compression Type - the configured compression type
Compressed Size - size in bytes of the log segment after compression
Space Savings - the reduction in size relative to the uncompressed size
Compression Ratio - the ratio of the uncompressed size to the compressed size
Avg Ratio/Batch - the mean compression ratio on a per-batch basis
Time - how long it took to compress all batches for the given compression type
Speed - the average rate at which the compression type is able to compress the log segment

Proposed Changes

`kafka-compression-analyzer.sh` aims to compress messages in the same manner a producer would and record the different in size of each batch. The tool sequentially iterates over each `RecordBatch` in a log file (similar to `kafka-dump-log.sh`), compresses it into a new MemoryRecords object for each compression type supported by Kafka, and records the size of the batch both before and after compression. Since the tool only compresses existing batches as they were written to the log file and does not merge or split them, the tool effectively measures the resulting log size as if compression were enabled across all producers, without any other producer configurations being changed (ex. `linger.ms`).

If a RecordBatch is already compressed in the log, the tool will decompress the batch and recompress it using the other compression types. This allows the tool to report the resulting size of the log as if all RecordBatches were to be normalized to use a single compression type.

Notes

- The shell script will run `kafka.tools.LogCompressionAnalyzer`, which contains the source of the tool
- There is precedent for read-only tools that operate on log files (i.e. `kafka-dump-log.sh`), any consequences of running this tool on a log file on a broker would be shared by those tools
- The tool does not spawn multiple threads
- The tool will likely consume an entire core while running
    - Consider copying the log segment and running the tool on a non-broker machine to avoid starving the broker of CPU

Compatibility, Deprecation, and Migration Plan

This proposal adds a new tool and changes no existing functionality.

Potential Future Work

There may be situations where it is not desirable for all batches to be compressed with a single compression type. For this reason, it may eventually be useful to provide a way to restrict the batches being compressed for the analysis. For example, it might be possible to exclude batches compressed with a certain compression type from being recompressed, only analyzing the remaining subset of the log. However, this can be implemented as a follow-up addition once better motivation for what mechanisms are needed and how they might work is available.

Rejected Alternatives

Another approach could be to run the tool as a consumer-like process that would fetch batches from the Kafka cluster and perform the compression measurements directly on those batches. This would require the tool to be provided the appropriate authentication information for the topic/partition being analyzed. This would also require batches of records to be exposed to the tool, which the consumer's interface and internals (specifically the fetcher) do not currently expose.