Kafka Log Compaction

Kafka Log Compaction Overview

Tos

- Recall Kafka can delete older records based on
  - time period
  - size of a log
- Kafka also supports log compaction for record key compaction
- Log compaction: keep latest version of record and delete older versions

Log Compaction

Tos

- Log compaction retains last known value for each record key
- Useful for restoring state after a crash or system failure, e.g., in- memory service, persistent data store, reloading a cache
- Data streams is to log changes to keyed, mutable data,
  - e.g., changes to a database table, changes to object in in-memory microservice
- Topic log has full snapshot of final values for every key - not just recently changed keys
- Downstream consumers can restore state from a log compacted topic

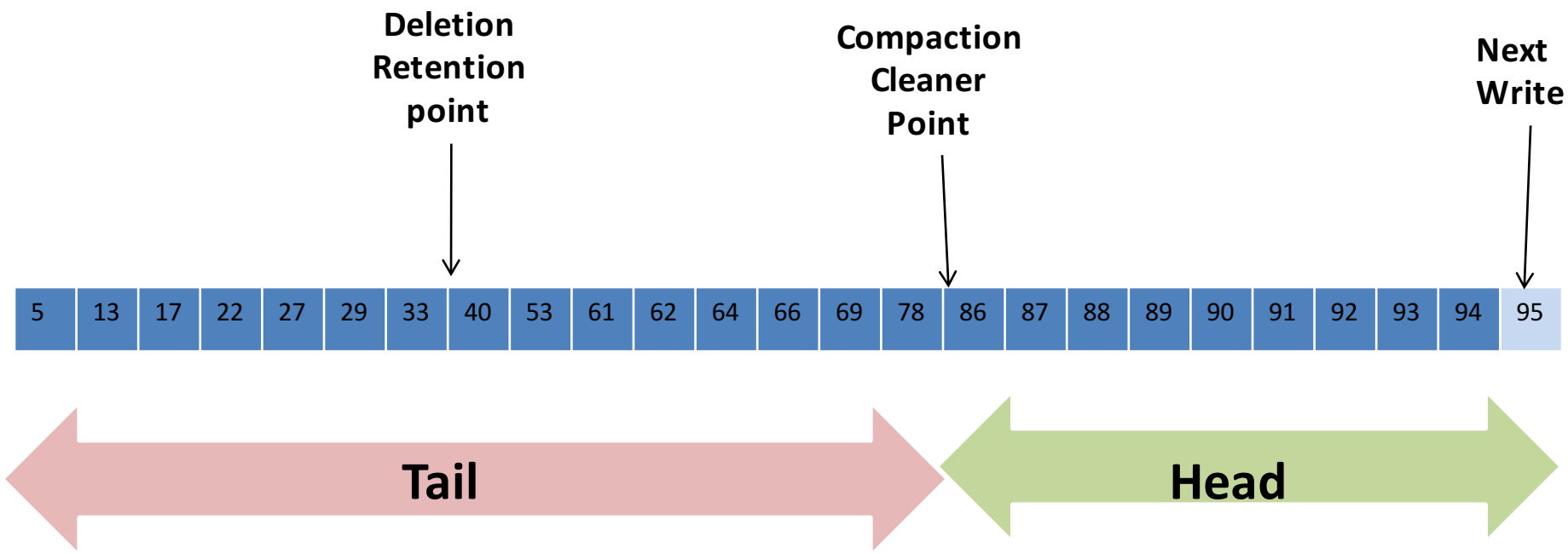
Log Compaction Structure

Tos

- Log has head and tail
- Head of compacted log identical to a traditional Kafka log
- New records get appended to the head
- Log compaction works at tail of the log
- Tail gets compacted
- Records in tail of log retain their original offset when written after compaction

Compaction Tail/Head

Tos



Log Compaction Basics

Tos

- All offsets remain valid, even if record at offset has been compacted away (next highest offset)
- Compaction also allows for deletes. A message with a key and a null payload acts like a tombstone (a delete marker for that key)
  - Tombstones get cleared after a period.
- Log compaction periodically runs in background by recopying log segments.
- Compaction does not block reads and can be throttled to avoid impacting I/O of producers and consumers

Log Compaction Cleaning

Tos

Before Compaction

Offset	13	17	19	20	21	22	23	24	25	26	27	28
Keys	K1	K5	K2	K7	K8	K4	K1	K1	K1	K9	K8	K2
Values	V5	V2	V7	V1	V4	V6	V1	V2	V9	V6	V22	V25

Cleaning

Only keeps latest version of key. Older duplicates not needed.

Offset	17	20	22	25	26	27	28
Keys	K5	K7	K4	K1	K9	K8	K2
Values	V2	V1	V6	V9	V6	V22	V25

After Compaction

Log Compaction Guarantees

Tos

- If consumer stays caught up to head of the log, it sees every record that is written.
  - Topic config ***min.compaction.lag.ms*** used to guarantee minimum period that must pass before message can be compacted.
- Consumer sees all tombstones as long as the consumer reaches head of log in a period less than the topic config ***delete.retention.ms*** (the default is 24 hours).
- Compaction will never re-order messages, just remove some.
- Offset for a message never changes.
- Any consumer reading from start of the log, sees at least final state of all records in order they were written

Log Cleaner

Tos

- Log cleaner does log compaction.
  - Has a pool of background compaction threads that recopy log segments, removing records whose key appears in head of log
- Each compaction thread works as follows:
  - Chooses topic log that has highest ratio: log head to log tail
  - Recopies log from start to end removes records whose keys occur later
- As log partition segments cleaned, they get swapped into log partition
  - Additional disk space required: only one log partition segment
  - not whole partition

Topic Config for Log Compaction

Tos

- To turn on compaction for a topic
  - topic config ***log.cleanup.policy=compact***
- To start compacting records after they are written
  - topic config ***log.cleaner.min.compaction.lag.ms***
  - Records wont be compacted until after this period

Broker Config for Log Compaction

Tos

NAME	DESCRIPTION	TYPE	DEFAULT
log.cleaner.backoff.ms	Sleep period when no legs need cleaning	Long	15,000
log.cleaner.dedupe.buffer.size	The total memory for log dedupe process for all cleaner threads	Long	134,217,728
log.cleaner.delete.retention.ms	How long record delete markers (tombstones) are retained	Long	86,400,000
log.cleaner.io.buffer.size	Total memory used for log cleaner I/O buffers for all cleaner threads	Int	524,288
log.cleaner.io.max.bytes.per. second	This is a way to throttle the log cleaner if it is taking up too much time	Double	1.797693134 8623157E3 08
log.cleaner.min.cleanable.ratio	The minimum ratio of dirty head log to total log(head and tail) for a log to get selected for cleaning	Double	0.5
log.cleaner.min.compaction.lag. ms	Minimum time period a new message will remain uncompacted in the log.	Long	0
log.cleaner.threads	Threads count used for log cleaning. Increase this if you have a lot of log compaction going on across many topic log partitions	Int	1
log.cleanup.policy	The default cleanup policy for segment files that are beyond their retention window. Valid policies are:”delete” and “compact”. You could use log compaction just for older segment files instead of deleting them, you could just compact them.		

Lab :

confluent-rebalancer- Auto Data Balancing
Movement of Partition from a Broker to another Broker.