Design Document - Chunky Logs

Requirements

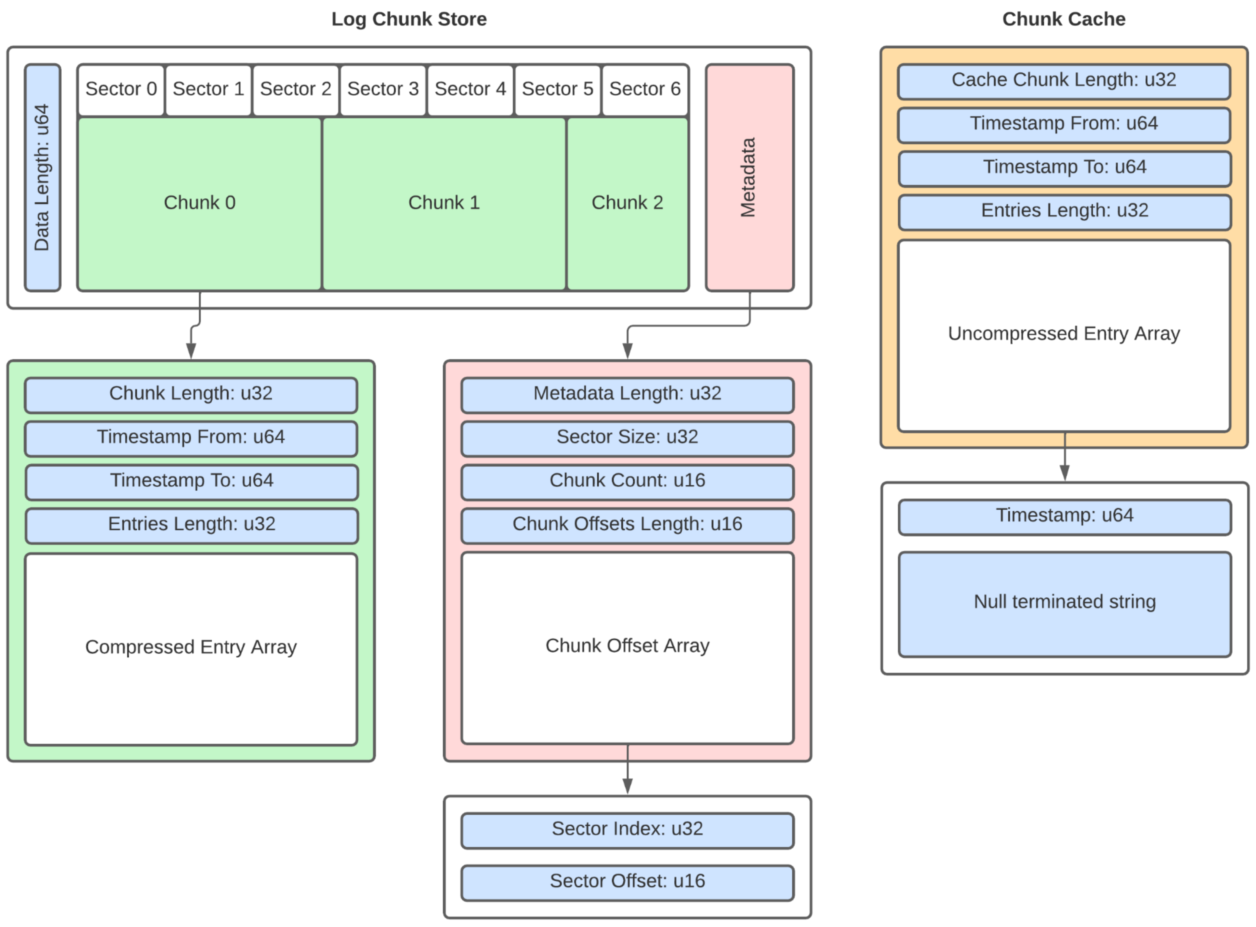
- Space efficient logging
- Cached writing for recovery
- Chunked compression

Overview

Storage format for high frequency writes for temporally linear data. Data is written in compressed chunks, each chunk has n data elements. Data is initially written to a cache which is periodically flushed to a cache file. Once the cache file is full it is compressed and written to the data store.

Details

Structure



Initialising

A new file is created with data length set to 0. After this the metadata chunk is written in default layout [1]. In addition to this, a new cache file is created in default layout [2].

Loading

Initially the data length is read from the chunk store, then a relative seek of the data length is performed from the current position. From there the metadata length is read and used to process the metadata section.

Next, if a cache file exists, the length is read and used to load it as a mutable memory map. If the file does not exist, a new file is created with a size of 64192 bytes (192B header + 64KB entry array) with default layout [2].

Caching

TODO

Writing

Cache Flush

Using one of the following

```
MmapMut::flush_async(&self) -> Result<()>
MmapMut::flush_async_range(&self, offset: usize, len: usize) -> Result<()>
```

Bulk Writes

TODO

Reading Chunks

TODO

Searching Chunks

TODO

Appendix

Default Layouts

[1] Log Chunk Store

- Metadata length: 96u32
- Sector size: 1000u32
- Chunk count: 0u16
- Chunk offsets length: 0u16
- Chunk offsets array: empty of 0usize footprint

[2] Cache

- Cache chunk length: 64192u32
- Timestamp from: 0u32
- Timestamp to: 0u32
- Entries length: 0u32
- Entry array: empty with 64KB memory footprint