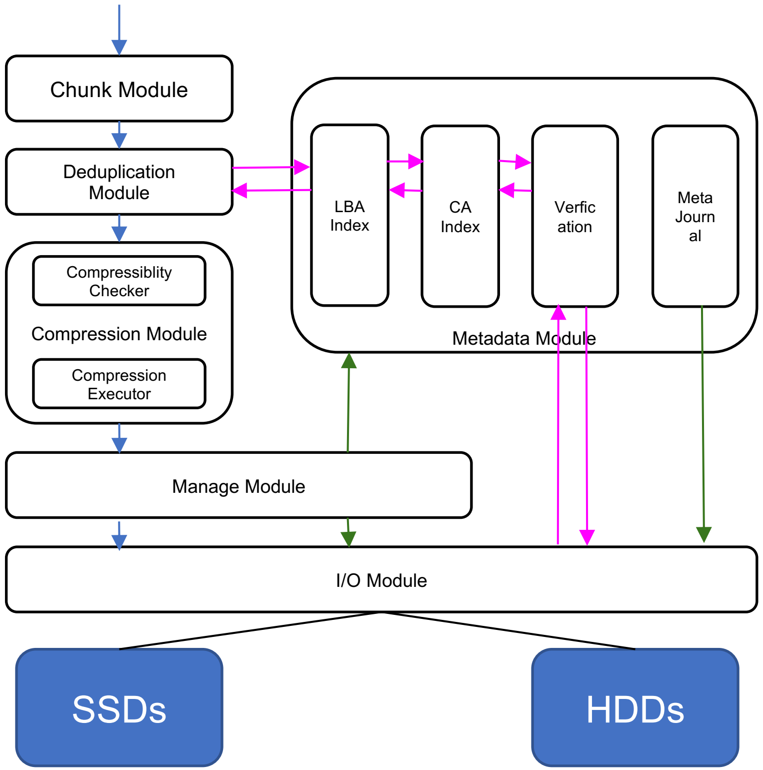
Architecture

2019.06.03

Overall Architecture



LBA: logical block address
CA: content address (fingerprint)

Chunk Module:
chunking incoming writes into (lba, length, data_buffer)

Deduplication Module:
checking whether incoming chunk is duplicate by consulting metadata module

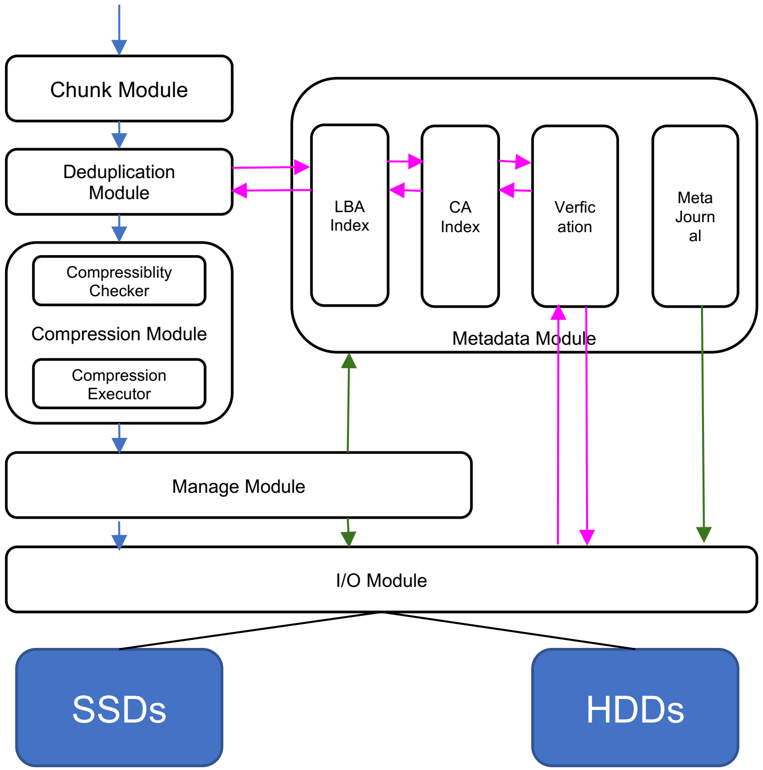
Compression Module:
including compressibility checker (checking whether chunk is compressible and decide whether to compress it or not) and compression executor (possibly integrate various implementation)

Manage Module:
organize data flow and communicate with I/O module

Metadata Module:
including indexes and journaling submodule
fetch metadata directly from I/O module to verfiy a true duplicate

Blue: Data flow, Pink: Metadata query flow, Green: Metadata update flow

Overall Architecture

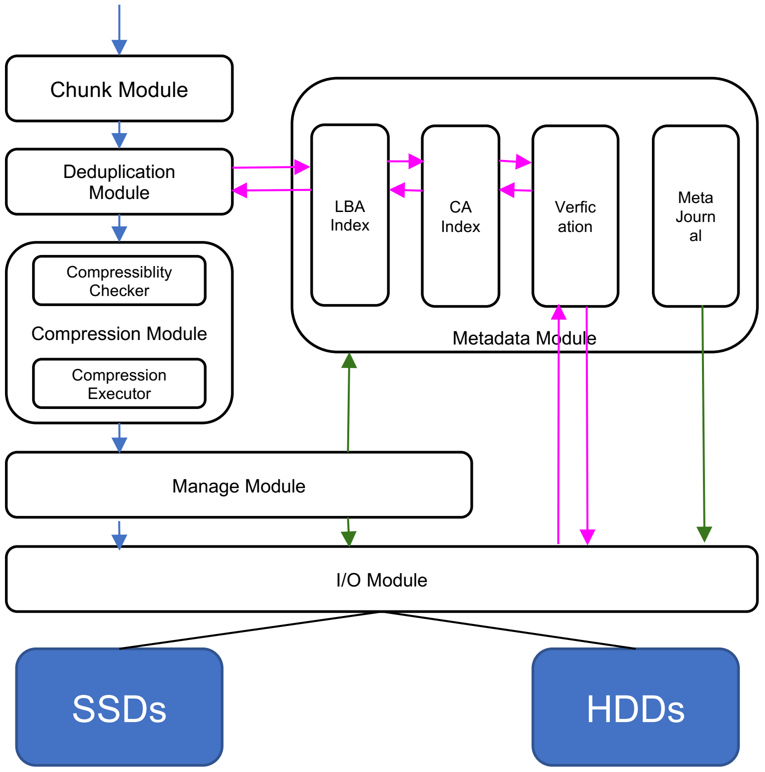


LBA: logical block address
CA: content address (fingerprint)

In and out of each module:
Blue (Data flow):
Chunk Module: in (lba, length, buffer), out (lba, buffer, ca) (8K size chunk)
Deduplication Module: in (*), out (*, ssd_location, duplicate_flag)
Compression Module: in (*), out (*, buffer_compressed)
Manage Module: in (*), out (*)
I/O Module: in (*), out for SSD (ssd location, buffer_compressed), out for HDD (lba, buffer)

Note:
Manage Module decides to write ssd or hdd according to the duplicate_flag.
Duplicate_flag == duplicate_write: write nothing
Duplicate_flag == duplicate_content: write only hdd
Duplicate_flag == not_duplicate: write both ssd and hdd

Overall Architecture

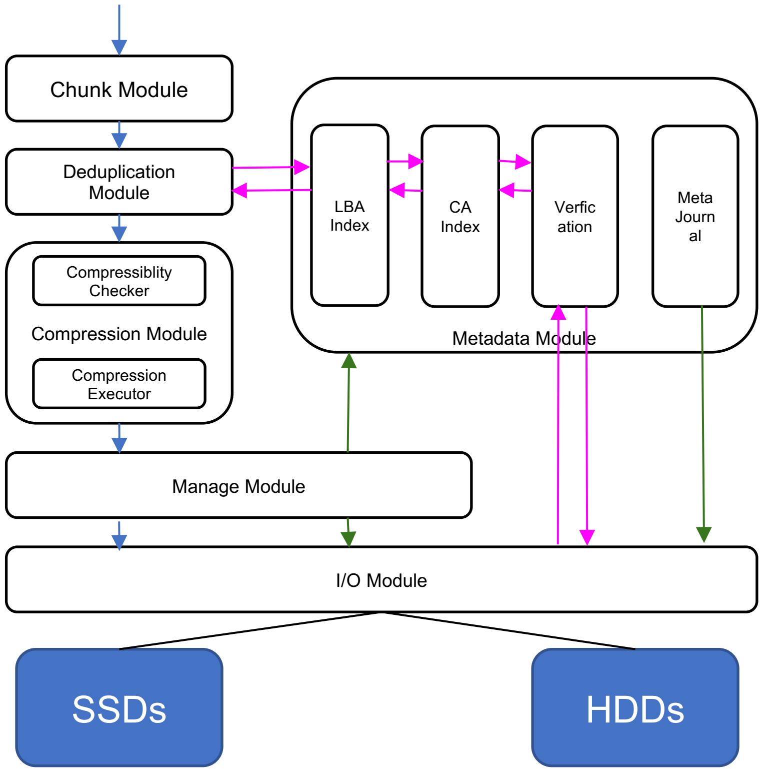


LBA: logical block address
CA: content address (fingerprint)

In and out of each module:
Pink (Metadata query flow): proceed from left to right then reverse

1. Deduplication Module: out (lba, ca)
2. lba Index: in (*), out (*, lba_match, ca_old)
3. ca index: in (*), out (*, ssd_location)
4. verfication: in (*), out (ssd_location), in (metadata), out (duplication_flag)
5. ca index: in (duplication_flag), out (duplication_flag)
6. lba index: *

Overall Architecture

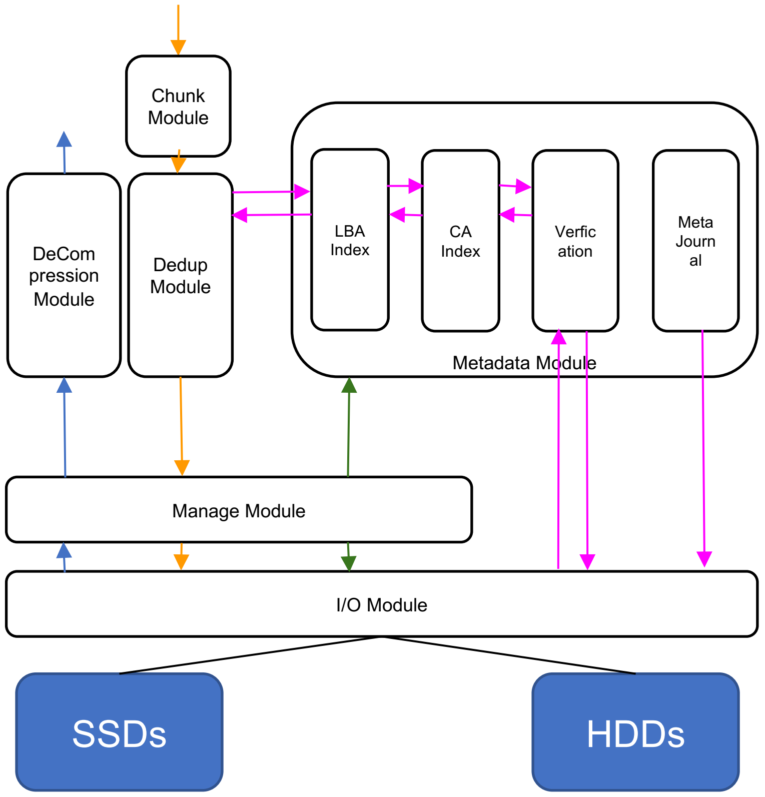


LBA: logical block address
CA: content address (fingerprint)

In and out of each module:
Green (Metadata update flow):
Manage Module: out for Metadata Module (lba, ca), out for I/O module (ssd_location, metadata)
Metadata Module: out for I/O Module (change log of the in-memory indexes)

Note: metadata stored in SSDs is either created newly or from the metadata querying process. In the second case, the metadata could be passing from deduplication module in the corresponding argument structure

Read Path



LBA: logical block address
CA: content address (fingerprint)

Orange: Request data flow, Blue: Response Data flow, Pink: Metadata query flow, Green: Metadata and data update flow

The processing is mostly the same as write path, except for the response part.

Note:
manage module will take care of caching recently read duplicate chunks. (Green path takes care for this process)