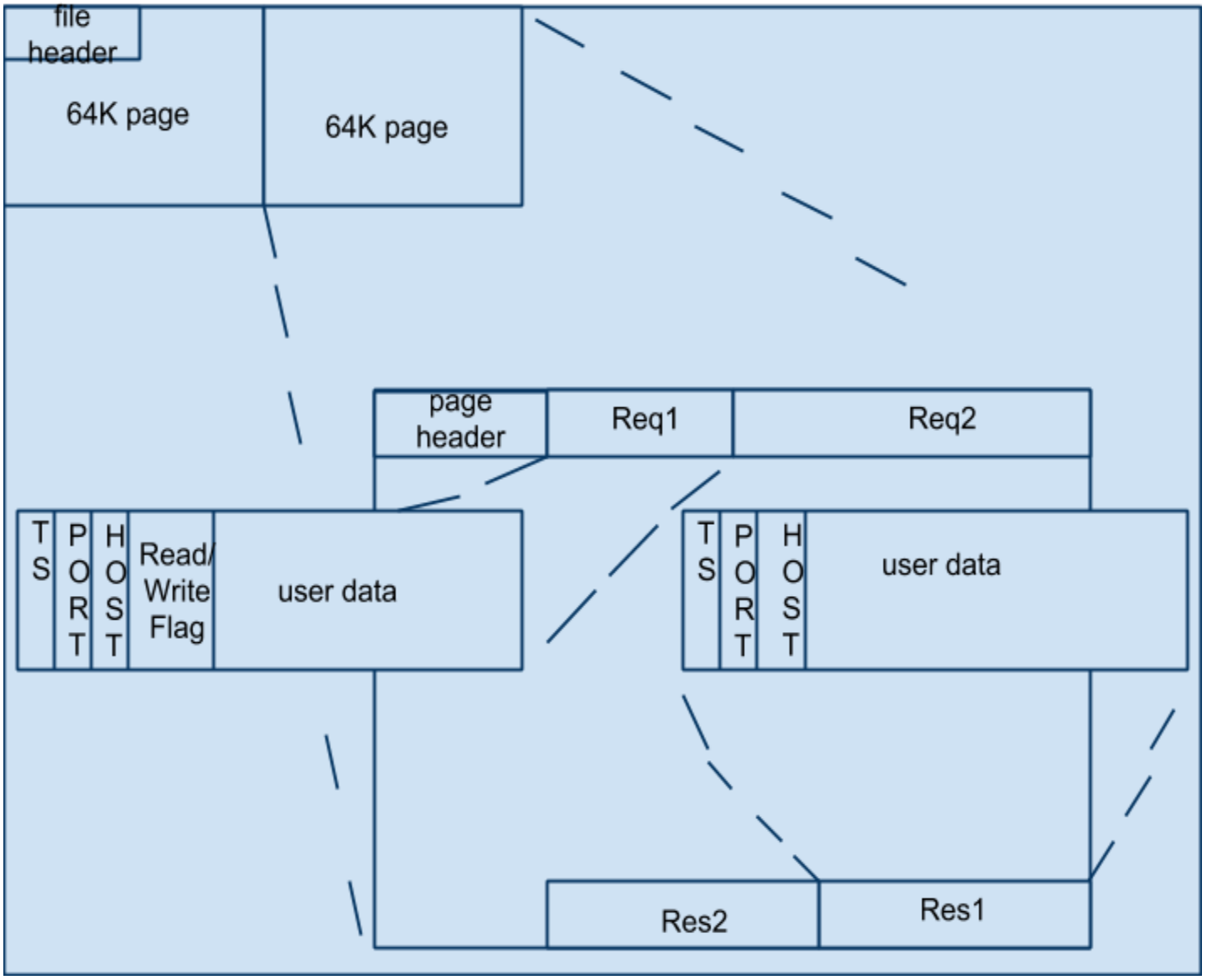
Galore-DB-High Level Design

Some of the requirements considered while designing the database for Galore.

- Each file should be self contained; meaning a file can be copied to another location and the replay agents should be able to read the entire information from the file to be able to run the reply.
- The file format should be platform independent
- Capture should be sequential, meaning, there should be minimum overhead while writing the captured data to disk
- Pulling out a response at a given timestamp, session/port should not be a full file scan.
- support data partitioning
- concurrency control; A file should be accessible to multiple threads/processes. However, as the database do not have to support transactions, this can easily be done by maintaining the current file offset for each file. The reader can always read the data before the current file offset.
- indexing; we can create separate index file(s) for the whole database.
- the size of the data stored on the disk should always be the minimum set of information needed to be able to re-create the data(this is not necessarily compression, but can be done with hash tables)
- The initial version should support a data size of 1TB.

Big Picture



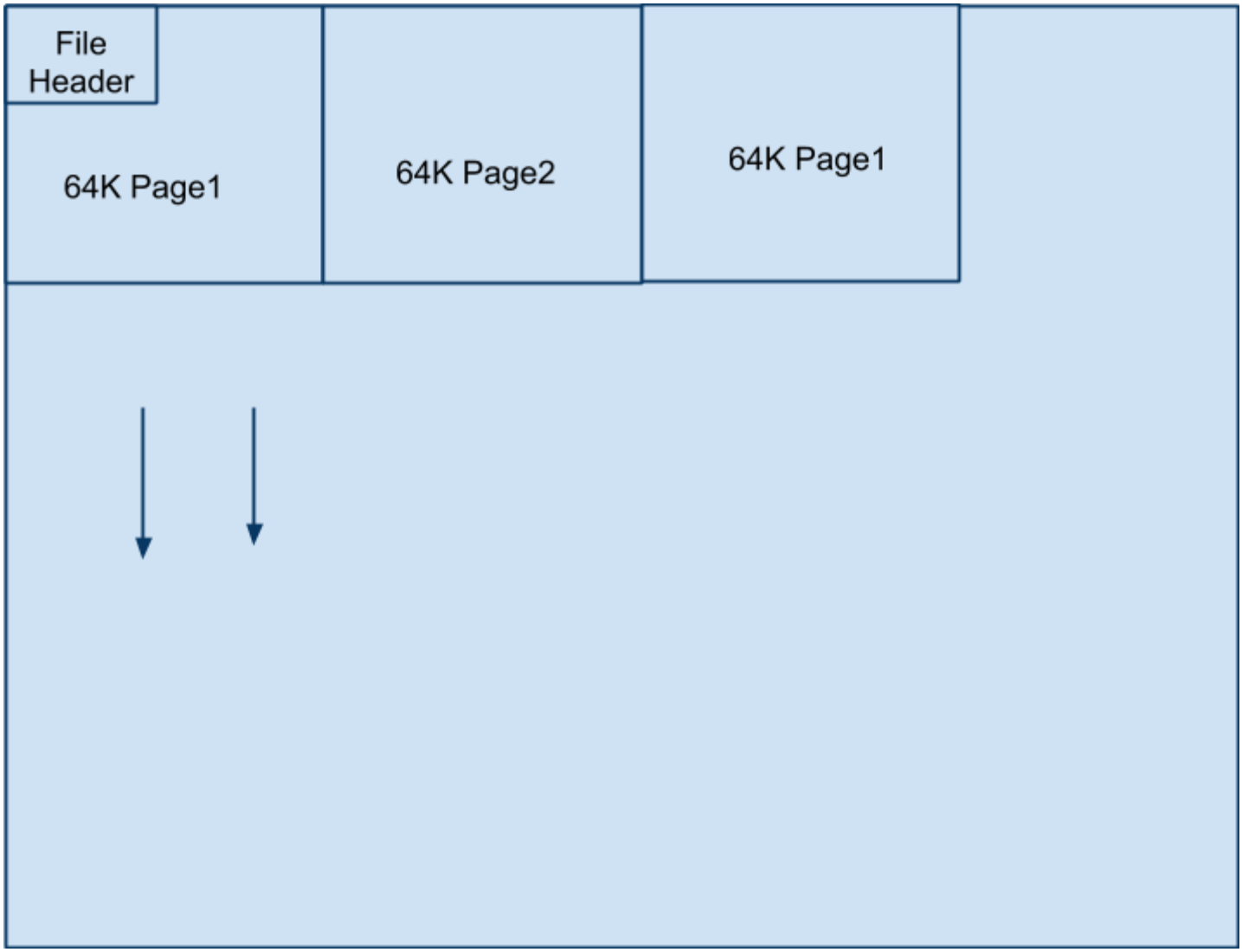
Data Layout on Disk

- All the captured data is stored on well formatted files on the disk. These file sizes can be arbitrary. However, we make sure that a session information is not captured across physical disk files. This way, we make sure that the file is self contained. the replay process never needs to look at more than one file
- size of the file is limited to a user defined size; however, the eventual size is some what close to the size specified. Once the file reaches the size specified, we will only write the data of the current sessions until they finish. however, any new sessions data will be written to another file.
- users with multiple disks, can have multiple partitions for captured data; in which case the captured sessions are equally balanced between the two disk files. However with the simple rule being, once a session is allotted a disk file, then the entire data related to the

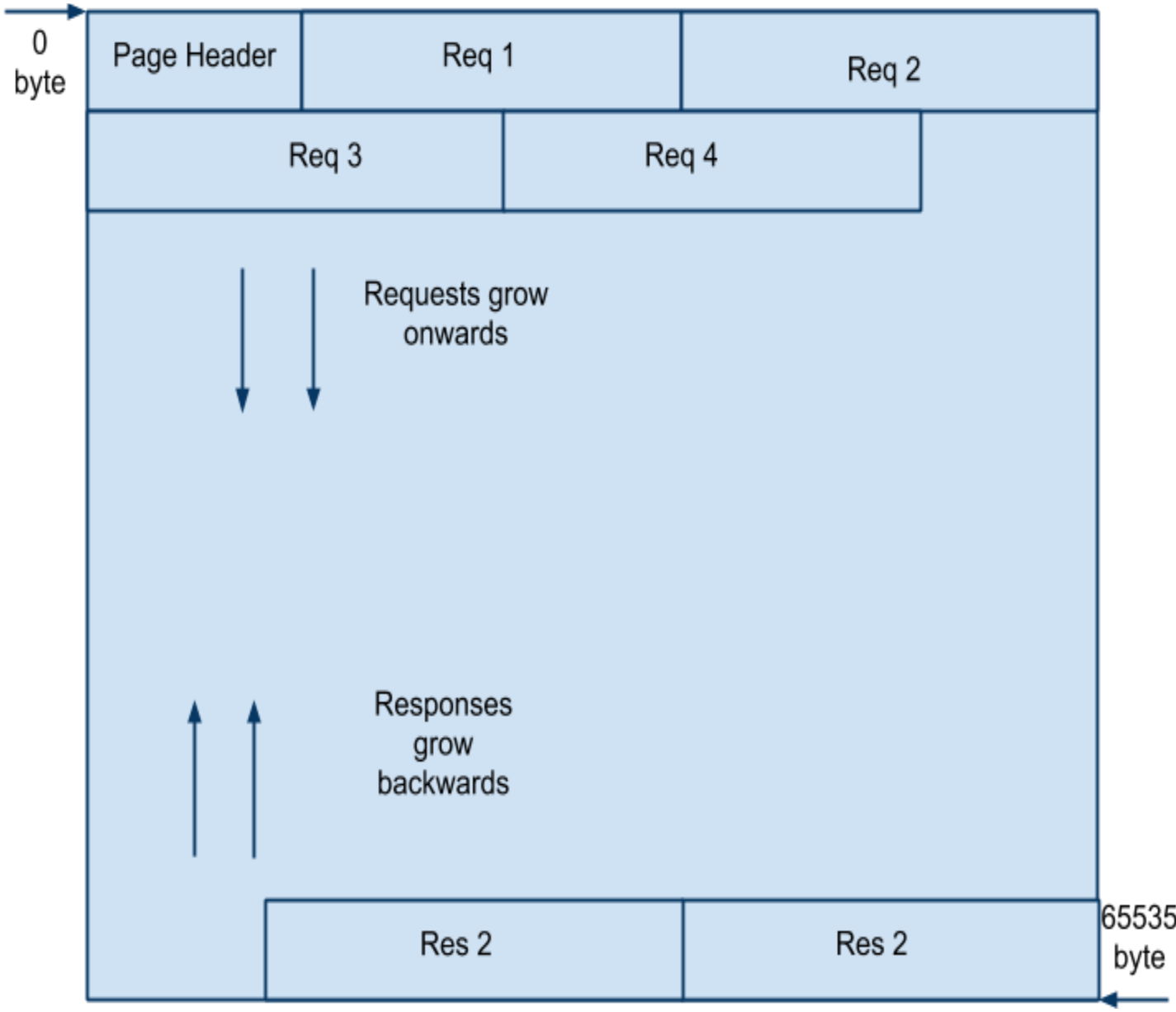
session will be in the same disk file.

- Every file has a file header, which stores information like
    - time stamp of first session start
    - time stamp of last session end in the file ; these are useful the replay process to look for the correct file(in case a single replay process is handling all the captured files); not only the replay process, it also helps the result viewer to pull the right response based on the time stamp. It does not have to go over all the files to pull the response
    - num_of_sessions
    - Index of time ranges for each of the 64K size page, the file consists of.
    - bit map tables, that stores simple strings to bits table.
    - file header should also contain the protocol of all the req/response pairs it is storing. this makes the multiple protocols go in multiple files. It brings down the complexity of storing by an order of magnitude.

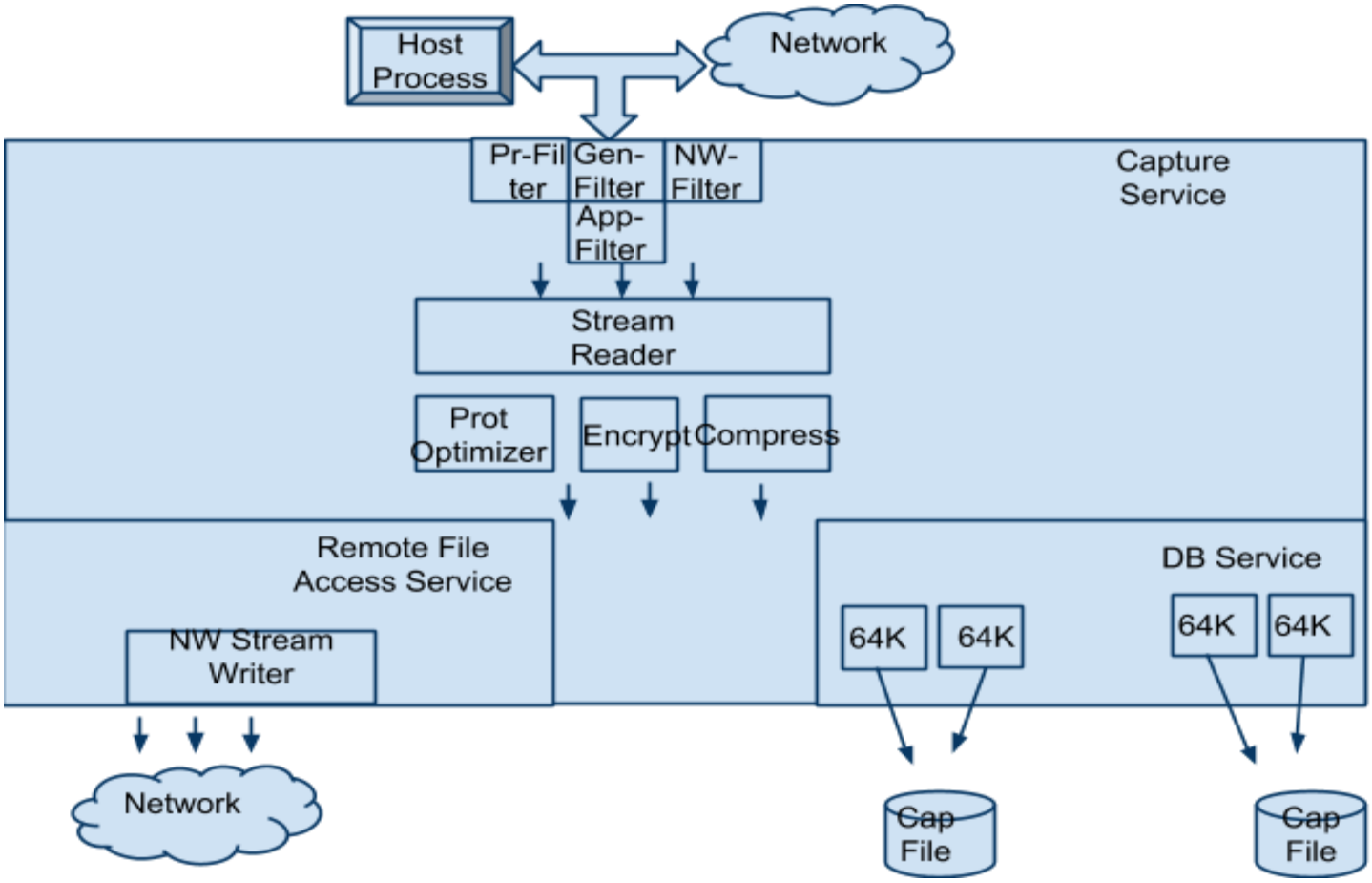
- The disk file is organized into chunks of 64k size pages. Every write to the disk file happens in the chunks of 64k size pages. This way we can make sure that we pool enough data before start writing to the disk.



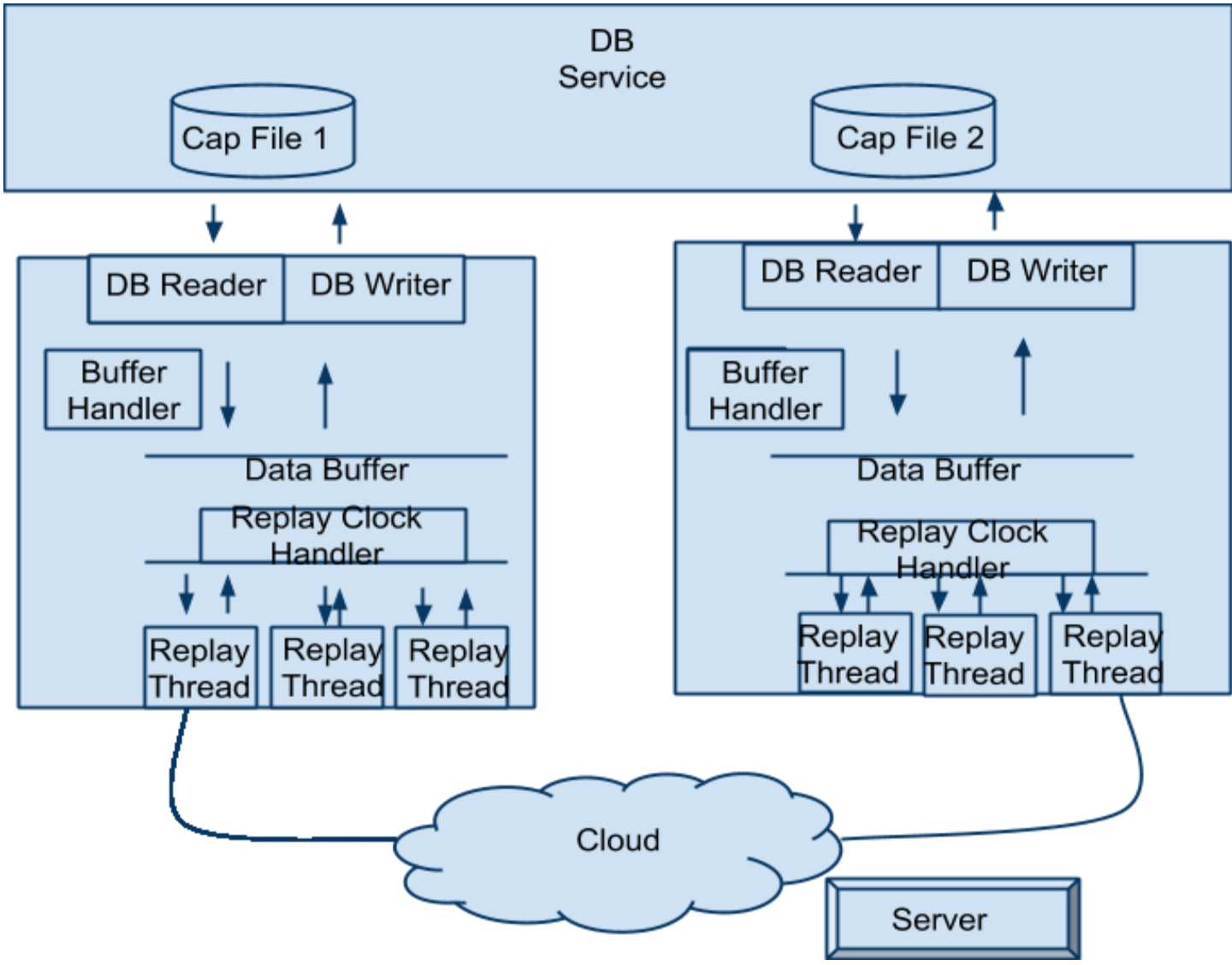
- Each 64k size page is organized into three sections.
    - page header; consists of information like
        - time stamp range;
    - Requests; all requests start from the end of page header and grow onwards.
    - Responses; all responses start from the end of the page and start growing backwards.
- page is considered full, when both the offsets of request writes/response writes cross.
- A request or response can span across the pages; this would ensure that there would not be holes in the data files.



Capture Service Interaction with DB



Replay Service Interaction with DB


DB
Service
Cap File 1
Cap File 2
DB Reader
DB Writer
Buffer
Handler
Data Buffer
Replay Clock
Handler
Replay
Thread
Replay
Thread
Replay
Thread
DB Reader
DB Writer
Buffer
Handler
Data Buffer
Replay Clock
Handler
Replay
Thread
Replay
Thread
Replay
Thread
Cloud
Server