(a file) into a plurality of logic blocks such as logic block 1, logic block 2, logic block 3, … logic block n as shown in FIG. 2 according to preset logical data block sizes (Block size), where n is a positive integer.

[85] In this way, the client may directly read corresponding data at a specified location of the data storage system according to logic blocks.

[86] Specifically, when reading data in the data storage system, the client may first determine which logic fragment (a logic block) of the target data (a file) is to be read according to a reading offset (position), and may then directly read, according to the returned metadata in Step S120, data at a specified location (e.g., an initial location of a file + a location corresponding to the offset) without further retrieving metadata services.

[87] On the other hand, after receiving logic blocks of target data from a client, the data caching system may also perform caching according to the logic blocks, for example, caching the same logic block on the same caching node, or caching different logic blocks of a same file (target data) on different caching nodes. The data caching system may maintain a table of the correspondence between the logic blocks of the file (target data) and cache nodes. For example, as shown in FIG. 2, for the file "oss://bucket/directory/file1", the correspondence between a plurality of logic blocks and cache nodes, e.g., <logic block 1, node 1>, <logic block 2, node 1> …, are recorded. Accordingly, the indexing relationships between respective logic blocks of a file and cache nodes may be established.

[88] In this way, when requesting for the target data again, the client may also determine which logic block of the target data (a file) is to be read according to the offset of data requested currently by the client in the target data, and perform reading directly at a corresponding cache node in the event of determining, based on metadata, that the cached target data of the data caching system is consistent with the target data stored in the data storage system.

[89] So far, the present disclosure provides a highly efficient data caching solution which is capable of automatically checking whether raw data in a data storage system is changed upon reading data, and caching and updating data efficiently, and may also perform blocked caching efficiently when necessary.