Symptom: If the existing file size is on the block boundary, immediate “seek()” call on opening such a file will cause an empty block being append at the end of the file. The process is illustrated as following:

1. A user calls FileSystem.append() on HDFS, it will in turn call DFSOutputStream.newStreamForAppend() to create an output stream for writing.
2. If the file size is on block boundary, DFSOutputStream will set the inner “DataStreamer” to PIPELINE\_SETUP\_CREATE state. That means, when new data comes, the data streamer will allocate a new block for the data.
3. However, what the user do is call seek() on the output stream, which cause the DFSOutputStream instance to issue a flush() call to the underlying layer to be prepared for moving the write cursor to elsewhere. The flush() call ask the underlying data streamer to finalize the block by sending an empty packet with “lastPacketInBlock” flag setting to true.
4. On receiving this packet, the data streamer will first allocating a block since it is in “PIPELINE\_SETUP\_CREATE” state, and setup the pipeline for writing. But this is just a flush call to ask the data streamer to close the current block…Then the data streamer will find this when it parse the packet. In effect, the data streamer create an empty block at the end of the file. For original append only HDFS, this is benign. However, this will cause problem when the file cursor is moved by seek() to the end of file, which will create another block at the end.

Possible solution:

1. On seek flush, if the data streamer state is still “PIPELINE\_SETUP\_CREATE”, we do not allocate new block only if there is no data to be written to the new block before the flush operation...
2. Think it over: is the normal flush the same?