Subject: U++: Why Archive Data?
Posted by aminhere on Mon, 20 Jun 2022 15:25:32 GMT
View Forum Message <> Reply to Message

Hi there,

While trying to learn from the source code, I noticed that the framework makes heavy use of compression/ decompression to read and write data.
As far as I had understood, compression provided the advantage of storing data in a more compact manner while still maintaining integrity.
But as I looked more into LZ4 algorithm in general, some sources mentioned that it provides faster read/ write than direct read and write. I am wondering why this is that case, because no matter what, the main data still has to be processed - the compression/ decompression is just another extra step. Even if we consider a basic compression algorithm like Huffman coding, we are still having to examine the original data space either way, so a regular read, for example would do just that. But a compression algorithm would not only have to perform that step, but also then have to process that information further. How could the presence of extra steps yield faster processing given that both a regular IO operation and compression/ decompression operation seem to be performing the initial data space read.
U++ mainly seems to use the zlib library heavily for writing/ retreiving app related resources. Is this done simply to use space efficiently, or for other reasons as well, like the one mentioned above?
Any help would be greatly appreciated.

Regards,
Amin