**Compilation Instructions**

cd to kanzi/java and run *mvn compile*

**Execution Instructions**

For the command line serialization and compression tool, an example command is:

*java -cp "target\classes;lib/\*" kanzi.app.Kanzi -c -i data/dataset.json -o output/comp.knz -f -t BWT -b 1m -e Huffman -j 1 --serializer=arraylist*

The flag -c or -d controls whether to perform compression or decompression respectively. Decompression cannot be used concurrently with serialization.

-i and -o denote the input and output files respectively.

-f forces the file to be overwritten if the output file already exists.

-t selects the transform to be used. Those we are using are BWT and LZ.

-b controls the block size, here 1m means 1 megabyte.

-e selects the entropy coding. For this we are only using Huffman.

-j selects the number of jobs to be run. This is set to 1 for all experiments to minimize the variability in multithreaded performance.

--serializer= selects a serializer to use on the input file before compression. Options available here are arraylist, hashfunction, hashgrid, hashmapchaining, and hashmap.

We gathered data on compression by running the command line tool from the script found in compressionTest.bat. Output for this is logged into log/data.csv.

We gathered data on serialization using the program in kanzi/serializer/Test.java. This can be run with the command: *java -cp "target/classes;lib/\*" kanzi.serializer.Test* Output for this is logged into log/log-output.txt.

**Attribution**

The compression and utilities provided by Kanzi are used as per the Apache license: <http://www.apache.org/licenses/LICENSE-2.0>

Modifications to the code were made to integrate with serialization, as well as to record and log data to csv. Code in kanzi/serializer is original.