Project 4: Dictionary Encoding and Query

Section 1: Encoding

Test	Number of Threads	Encoding Time (s)	Percentage
1	1	116.2	100%
2	2	90.3	77%
3	4	64.8	56%
4	8	54.6	48%

Table 1: Encoding Time Test Result Using Various Number of Threads

While increase the number of threads can lead to a notable increase in speed. Yet the effect is not as significant as expected. For test 1-4, the gain increase linearly but the number of threads doubles each time. Increasing the threads from 4 to 8 gets a much smaller gain One possible explanation is that the encoding process involves a big overhead, so the effect of parallelism on the actual encoding process is attenuated. Another reason could be that the OS did not assign the resources to the program as it requested, especially when the number of threads become larger.

Section 2: Query

Although Python does not natively supports SIMD, The Numpy library is highly optimized using SIMD. Therefore in this project I will use Numpy functions to serve as SIMD optimization [1]

Query Request	Optimization Enabled	Optimization Disabled	Vanilla
Item: knvr	0.13	3.8	3.8
Prefix: knv	5.5	171.1	15.9

Table 2: Query Time (in seconds) Test Result

As one of the most common lossless compression method, dictionary coding can reduce the size of the dataset from 1.0G to 633MB.

Without optimization, the dictionary does not show any advantage over the vanilla approach in terms of speed. This is because both need to iterate through the set in a for loop. Another reason may be that python w/o librarys is not good at speed and it takes about the same time to compare strings and integers. However, when optimization like SIMD is enabled, the time needed for search/scan is significantly reduced.

Note that for vanilla scan query, there is a method named “string.startwith(prefix)” which can quickly determine if a string has the prefix. Therefore the time needed for each iteration is very short (only one comparison is required). For dictionary encoding, since the data are now integers, multiple comparisons must be done in each iteration. Without any special functions or optimization this can be very time-consuming. This result point out a drawback in dictionary encoding: the coding will distort the features of the each data (such as prefix) so any scans that are based on these features will be slow. Optimization are needed to maximize the benefit of dictionary encoding.

Reference

[1] https://numpy.org/doc/stable/reference/simd/index.html