Labview course - Lesson 10/05/2018 - Dott. Andreotti Mirco

Statistical analysis of simple Labview wait (ms)

In these examples we find the right implementation of a Labview application used to make a statistical analysis of the function wait (ms) available in Labview.

1. **wait_study_1.vi**

    a. In a single while loop the time difference between a cycle and the previous is implemented with the shift register tool.
    b. Delta_t is then appended to a cumulative vector.
    c. Delta_t is plotted in a waveform chart.
    d. Cumulative Delta_t vector is elaborated with GeneralHistrogram labview tools to produce histrogram of values.

    Running the vi it appears that Delta_t increases with time, because maybe cumulative vector, waveform chart increase the amount of data and GeneralHistogram has to elaborate increasing data.

    This is not the right solution to analize wait function, because the implemented structure affect the execution time of a loop.

2. **wait_study_2.vi** and
3. **wait_study_3.vi**

    a. Two parallel while loop are implemented (a producer and a consumer).
    b. In the producer loop Delta_t is calculated and appended to cumulative vector and sent to waveform chart.
    c. In the consumer loop the cumulative vector is analized with GeneralHistogram tool.

    Running the vi it appears that Delta_t increases with time, with different speed respect before. The elemets that could affect the loop time are cumulative vector and waveform chart increasing the amount of data.
    In wait_study_3 just the waveform chart is deleted, but Delta_t still to increase, then the strategy to accumulate data into an increaing vector affects the execution loop time.

4. **wait_study_4.vi**

    a. Two parallel while loop are implemented (a producer and a consumer).
    b. A queue for Delta_t is implemented in place of cumulative vector.

c. In the producer loop Delta_t is calculated and inserted to queue, and the application take care of queue overflows.
d. In the consumer loop the queue is flushed and appended to cumulative vector of Delta_t, and this is analized with GeneralHistogram tool.

Running the vi, for a first period of time it seems to work fine, no overflows happens. In the follow overflows starts growing up, because cumulative vector are increasing, slowing down the consumer loop. In this situation the consumer loop is not able to follow producer loop.
It is clear that the bottleneck of this implementation is the cumulative vector and GeneralHistogram tool.

5. wait_study_5.vi

a. Two parallel while loop are implemented (a producer and a consumer).
b. A queue for Delta_t is implemented in place of cumulative vector.
c. A frequencies (entries) vector is implemented in place of an histogram output.
d. In the producer loop Delta_t is calculated and inserted to queue, and the application take care of queue overflows.
e. In the consumer loop the cumulative vector is not used, the queue is flushed and the returned vector is analized with GeneralHistogram tool.
f. From GeneralHistogram tools the frequencies vector of the restricted sample returned by GeneralHistogram is summed to general frequencies vector.
g. An X-Y graph is plotted with classes as X-vector and frequencies as Y-vector. The graph is set to appear as an histo.

This implementation allows to view the histogram of Delta_t updating as an online monitoring. Since the system is not accumulating data no slowing down will happens, also running for long time.
The only limitation of this choice is that the classes of histogram cannot be changed programmatically during the execution.
In order to have raw data for offline analysis, a storage on file of raw data could also be implemented, avoiding a bottleneck in writing file procedure.