TCT AEROSPACE

QuadPace test

Test id: tct4mx102
Date: 12/05.2021
Iteration: V1.3

Test parameters:

- Evaluating processing performance of QuadPace binary code format as compared to standard binary format.
- Establishing the integration functionality of QuadPace binary code format, within a standard binary coding format environment.
- Establishing QuadPace can increase the efficiency of standard binary code format programs and by what factor it does so.
- Establishing that QuadPace is not solely a stand-a-long binary code format.

Test setup:

programs:
This test will consist to two computer programs, both designed and constructed within a C# code format.

1. Program ‘one’ will be titled ‘Binary version’. This program will contain no elements of QuadPace binary code format. It will represent a fully functional stand-alone C# program.
2. Program two will be titled ‘QuadPace’. This program will be constructed in C# but contain embedded elements of QuadPace binary code format, these elements will run as an internal QuadPace program within the C# program, resulting in a hybrid C#-QuadPace program.

3. Both programs will be designed to carry out exactly the same function in relation to a desired task. Both will carry out the function in an identical way, with identical parameters and settings.
4. Both programs are constructed from identical C# coding, with the exception that the program ‘QuadPace’ also contains elements of QuadPace binary code format.

Hardware:

Both ‘Binary version’ and ‘QuadPace’ programs will, for the purposes of this test, run on the same computer. No alteration will be allowed to any of the hardware before, during or after the testing period.

Hardware basics declared as;
Processor: AMD Athlon(tm) II X4 640 Processor 3.00 GHz
Internal Ram: 12.0 GB
System: 64-bit operating system, x64-based processor
OS: Windows 10
OS version: 20H2

Test procedure:
Both ‘Binary version’ and ‘QuadPace’ programs will run the same task procedure in the same configuration.

When initiated;
1. program will load the full text from the novel “War and Peace”
2. Program carries out a word count procedure and displays the result.
3. Program asks for one or more text inputs, representing words to be searched for in the full text (this input is case sensitive). To indicate that the operator has finished entering their word search text list, they enter ‘0’ as the last entry. All inputted words and ‘0’ will be displayed.
4. Program asks for a numeric input for how many times this search function should be carried out as a consecutive procedure. Numeric value is displayed.
5. Program, on completion of the above consecutive procedure, displays the time in milliseconds the search task has taken, each word searched for and the resulting count of how many times each word has appeared in the full text.

There is no limit on the number of text entry requests an operator can input, in respect to the word search. There is no limit to the numeric value an operator can input, in respect to the number of times the search function can be set for its consecutive procedure.

Declaration;

The search procedure is purposely designed to be CPU resource heavy, with each individual word searched independently before the next word is searched.

Example: In this test (tct4mx102) we have used five words to be searched, (there, cat, house, end, more). If we set the consecutive procedure value to '1', the program will first search the entire text of "war and Peace" just for the word 'there', once the 'there' search has been completed, the program will move on to search the entire text for the word 'cat', and continue this process for each individual following word inputted. If the consecutive procedure value is higher, for example 999, the program will repeat the above process as a loop until it has cycles through 999 full searches.

Results:

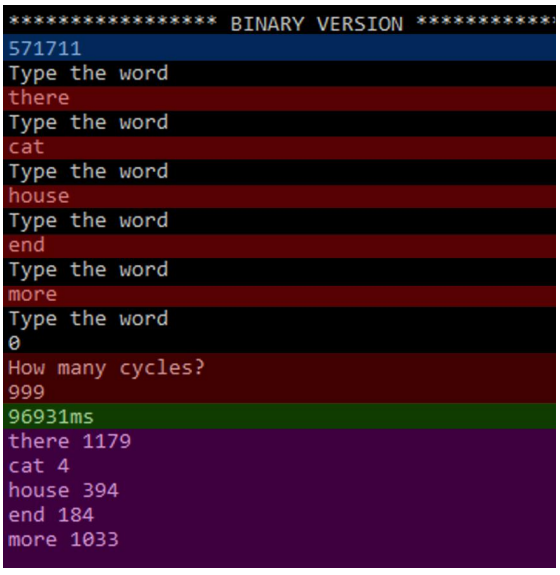

Fig 1

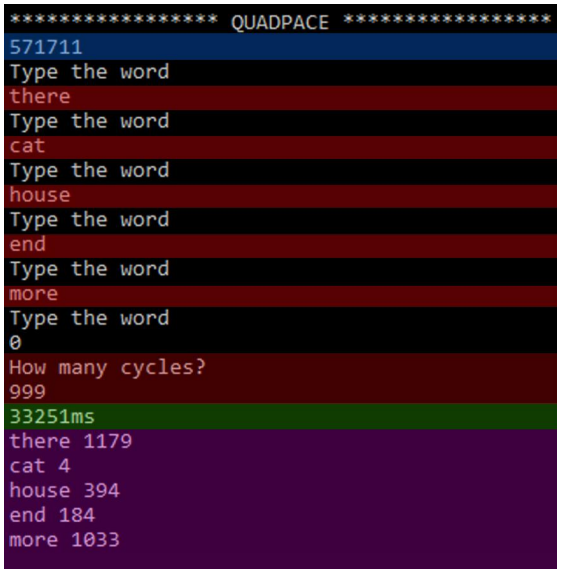

Fig 2

Fig one shows a screen grab of the output results of the test for 'Binary version', with Fig 2 showing a screen grab of the output results for QuadPace. Both indicate that the word count procedure carried out produced an identical result of 571,711 words. Both show the same word search inputs of "there, cat, house, end, more", with a consecutive

procedure (cycles) set for 999. Both programs output identical results with reference to how many times each individual inputted word appears in the full text.

Both programs performed the identical task with identical parameters, input and settings, however there is a clear discrepancy between the two programs for the time taken to perform the processing, with ‘Binary version’ having taken 96931m/s (96.9 seconds) to process the data and ‘QuadPace’ having taken just 33251m/s (33.3 seconds) to complete the data processing.

The above information indicates that ‘QuadPace processed’ the data at a speed 2.9 times faster when compared to ‘Binary version’.

Test parameters conclusions:

- Evaluating processing performance of QuadPace binary code format as compared to standard binary format. ***(QuadPace outperforms standard binary)***
- Establishing the integration functionality of QuadPace binary code format, within a standard binary coding format environment. ***(QuadPace can be fully integrated and function at 100% functionality)***
- Establishing QuadPace can increase the efficiency of standard binary code format programs and by what factor it does so. ***(QuadPace increased the efficiency of the C# code format by a factor of 2.9x)***
- Establishing that QuadPace is not solely a stand-a-long binary code format. ***(QuadPace is shown to be both a stand-a-lone binary code format and one which can be fully integrated with standard binary code formats)***