Assignment 3: Timing methods- An Analysis

Stephen Chambers

March 9, 2015

1 The Question

The goal of this assignment is to find and evaluate methods of precise timing in Linux/Unix environment. This paper aims to find and describe a few methods to determine how long an operation took. It will discuss the precision of timing that is expected to get using the method.

2 Introduction

The language chosen for this assignment is Python. The two methods being analyzed are the *time* and *timeit* modules. These modules were chosen because they are both are widely used and easily accessible ways to benchmark performance.

Although the timeit module is actually based on the *time.time()* function, it is widely regarded as the best choice in performance timing experiments. This is due to the fact that it disables the Python garbage collector before running.[1] This paper will aim to see if that makes a notable difference in timing precision.

3 Experiment

The function *time.sleep()* was used in order to provide an operation with little variance. Any errors that were uncovered would then be due to the timing methods themselves. Each experiment recorded the mean and standard deviation of all time values of a test run. The amount of time values generated during a test run varied exponentially in order to determine if more test runs correlated to a higher degree of accuracy. These values can be seen in in section 4.

For each module, the experiments were run with three different operations, explained in the Table 1 below.

[1] Reference: https://docs.python.org/2/library/timeit.html

Operation	Meaning
time.sleep(1)	Suspend execution for one second
time.sleep(0.1)	Suspend execution for 100 milliseconds
time.sleep(0.0001)	Suspend execution for 100 microseconds

Table 1: Operations performed for timing precision analysis

4 Results

This section shows the result of the experiment for both timing modules. Analysis of these results is in section 5. All results were rounded to four decimals.

4.1 Time module

Number of times run	Mean time(s)	Standard deviation(ms)
1	1.0030	0
2	1.0014	0.6205
5	1.0039	1.6605
10	1.0028	2.3177
20	1.0039	1.6675
50	1.0044	1.4306
100	1.0032	2.2646

Table 2: Results for the time module with the *time.sleep(1)* operation

Number of times run	Mean time(ms)	Standard deviation(ms)
1	105.1240	0
2	102.6320	2.4900
5	104.5787	1.1139
10	104.0043	1.7983
20	104.4622	1.6173
50	104.8844	0.7715
100	104.6774	1.1553
1000	104.6283	1.2865

Table 3: Results for the time module with the *time.sleep(0.1)* operation

Number of times run	Mean time(μs)	Standard deviation(μs)
1	204.0863	0
2	185.4897	15.4972
5	184.3452	15.7359
10	182.3187	17.7282
20	156.5695	19.4461
50	137.2385	11.3656
100	138.6714	15.1702
1000	139.8988	12.8314

Table 4: Results for the time module with the *time.sleep(0.0001)* operation

4.2 Timeit module

Number of times run	Mean time(s)	Standard deviation(ms)
1	1.0034	0
2	1.0047	0.4562
5	1.0029	1.9794
10	1.0044	1.4693
20	1.0040	1.9233
50	1.0037	1.9249
100	1.0039	1.8976

Table 5: Results for the timeit module with the *time.sleep(1)* operation

Number of times run	Mean time(ms)	Standard deviation(ms)
1	104.0690	0
2	104.7680	0.3381
5	104.5787	1.8027
10	103.4664	2.2441
20	103.0741	1.7988
50	103.9360	1.7860
100	103.9404	1.8695
1000	104.6494	1.2273

Table 6: Results for the timeit module with the *time.sleep(0.1)* operation

Number of times run	Mean time(μs)	Standard deviation(μs)
1	202.8942	0
2	182.9863	18.9543
5	187.2063	22.8874
10	181.0074	14.2101
20	172.5912	11.8834
50	131.8932	6.1943
100	139.4486	15.7386
1000	138.0274	10.0955

Table 7: Results for the timeit module with the *time.sleep(0.0001)* operation

5 Discussion

The table below states the 95% confidence intervals of each module based on a second, millisecond, and microsecond scale.

Module	Scale	Confidence Interval
time	seconds	1.0039 +- 0.00038
time	milliseconds	104.6283 +- 0.07616
time	microseconds	139.898 +- 0.79625
timeit	seconds	1.0039 +- 0.00038
timeit	milliseconds	104.649 +- 0.07616
timeit	microseconds	138.0274 +- 0.62647

Table 8: Results for the time module with the *time.sleep(0.0001)* operation

There was virtually no difference between the time module and the timeit module in regards to how precise and accurate the measurements were on different scales. Both modules can effectively measure time precisely in seconds, milliseconds, and microseconds with the above confidences. Based on the above table, both time modules had approximately 38 microseconds of overhead.

6 Conclusion

Either of the timing modules discussed in this paper will work effectively for second and millisecond measurements. They will not work effectively for microsecond measurements, as the overhead of setting up the timer affects the results.