Evaluation of precise timing methods

Varsha Thirumakil
Department of Computer Science
Assignment-3
vt2000@wildcats.unh.edu

Abstract— ***Computers and networks are becoming increasing fast, accuracy expectations are extending from milliseconds to nanoseconds. Hence the precision timing methods are analyzed widely throughout the computer networks inorder to deal with clock synchronization (the idea that internal clocks of several computers may differ). In this paper, various methods of precision timings in Linux/Unix environment are examined and results are evaluated. This has been acheived by carefully conducting an experiment, performing statistical analysis and quantitative comparisons of the timings.***

Keywords— Precision timing methods, time

I. DESCRIPTION OF THE EXPERIMENTS

In practical computing, wall time is the actual time, usually measured in seconds, that a program takes to run or to execute its assigned tasks. I have demonstrated few methods for determining the precision timing (wall time) for calculating set of prime numbers ranging from 2 to 10,000 in linux or unix envoronment.
They are :
1) Stopwatch method - Google Guava for Java : The **Google Guava** is an open-source set of common libraries for java. The nice thing is that Stopwatch.toString() does a good job of selecting time units for the measurement. This can clearly depict timings extending from milliseconds to nanoseconds.

2) **$SECONDS - Shell Script :** Apart from using the time command, the bash-specific magic built-in variable **$SECONDS**, which contains the number of seconds since the script started executing. Magic variable produces the results in seconds.

3) **time – Python :** time.time returns the wall clock time in seconds. Python's timeit module provides a simple way for timing and its less error prone. Besides calling in the program they can be used from the command line.

II. DETAILED SOLUTION ANALYSIS

A) Experiment :

This is a straightforward framework of evaluation of precise timing methods that consists of three classes- TimeTesting.java, primepy.py and prime.sh which runs a test for a set of 10,000 samples.

B) Experiment setup:

This experiment is designed in such a way that we use three timing methods and analyze which one closely generates precision/accurate timings. The results generated from each of the timing methods are carefully evaluated for 20 observations.

C) Measures:

Margin of error, Average, Standard Deviation and confidence intervals are some of the measures that have been used for evaluation of precision timings.

III. QUANTITATIVE COMPARISONS

<u>Tool used : SAS JMP</u>

I have utilized **variability chart** with mean diamond to represent the precision timing for different methods. The top and bottom of each diamond represent the 95% confidence interval for each group. The mean line across the middle of each diamond represents the group mean. I have also represented a range bar and individual data points are jittered.

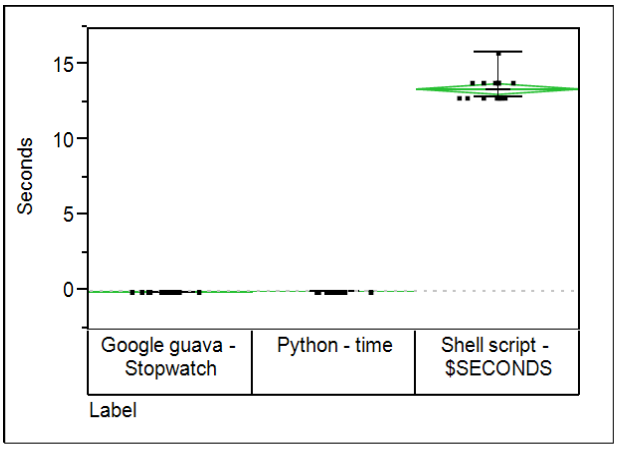


Figure 1 : Comparison of the timing methods

From figure-1 we can derive that the timing methods of Stopwatch class (Google guava) generated the results more effieciently compared to Pythons time, timeit methods and Shell Scripts $SECONDS variable. Guava's stopwatch class is a simple class that provides a method to conveniently measure time elapsed between two code points.

Stopwatch(guava) seconds	Shell script in s	Python in s
0.05444	14	0.07593
0.06506	14	0.07008
0.04445	13	0.07
0.03909	13	0.06878
0.04447	13	0.064
0.05154	13	0.06655
0.04083	13	0.0649
0.04291	13	0.06535
0.04767	13	0.07271
0.04214	13	0.0743
0.04099	14	0.06966
0.03814	13	0.06958
0.04337	13	0.07099
0.04875	13	0.07184
0.04944	13	0.07082
0.04825	16	0.07157
0.05383	14	0.07066
0.04725	14	0.07194
0.04773	14	0.0725
0.0498	14	0.06437

Table 1: Tabulated results of stopwatch vs $SECONDS vs time

From Table 1 we can determine the variability in timing of the methods used in java vs that of other methods. It can be seen that the errors are caused by the variable overhead of the calls. Longer tasks demonstrated that the magnitude of the error remains the same. I have shown the variability chart that compares the Stopwatch methods vs the python time module.The results where pretty close to each other.

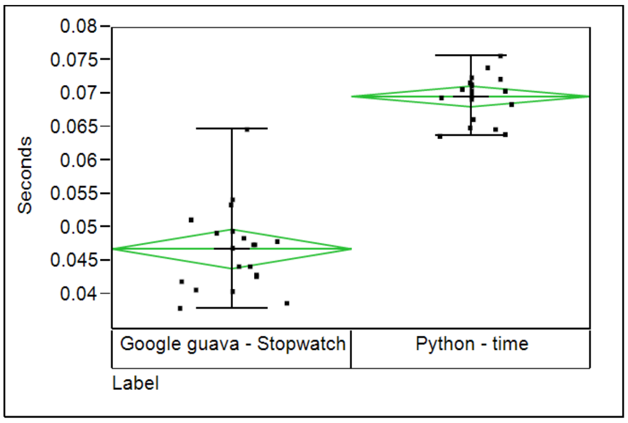
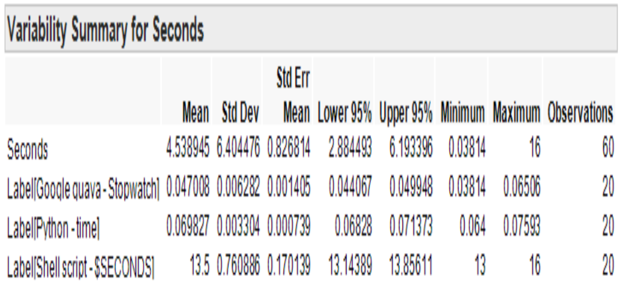


Figure 2 : Comparison of stopwatch vs time

Apart from testing the prime numbers for 10,000 samples, I also ran experiments to test the accuracy for reading the **scientific data**, reading and **writing csv files of huge data.** I came up with timings that were extremely faster to other timing methods.

IV. STATISTICAL ANALYSIS

From Table 2: we can say that mean average for the stopwatch class is the least and hence values are generated accurately in a faster time as well. The **margin of error** is a statistic expressing the amount of random sampling error in a survey's results which appears to be very less in case of the java precision timing method. Hence after conducting several trials we can say that shell script timing methods are very slow and have a significant margin of error in comparison with the other two methods. Java timing methods appears to be a little better compared to the python timing methods for generating wall clock time in seconds.



Variability Summary for Seconds

	Mean	Std Dev	Std Err Mean	Lower 95%	Upper 95%	Minimum	Maximum	Observations
Seconds	4.538945	6.404476	0.826814	2.884493	6.193396	0.03814	16	60
Label[Google quava - Stopwatch]	0.047008	0.006282	0.001405	0.044067	0.049948	0.03814	0.06506	20
Label[Python - time]	0.069827	0.003304	0.000739	0.06828	0.071373	0.064	0.07593	20
Label[Shell script - $SECONDS]	13.5	0.760886	0.170139	13.14389	13.85611	13	16	20

TABLE 2 :VARIABILITY SUMMARY

CONCLUSION

Thus an experiment for evaluation of precise timing was conducted, datas were collected and hence statistical analysis was performed. It can be concluded that more efficient timing method we use the more accurate the operation takes place. These experiments played a vital role to find out which precision timing method was beneficial.

REFERENCES

[1]http://docs.guavalibraries.googlecode.com/git/javadoc/com/google/common/base/Stopwatch.html

[2] http://www.tutorialspoint.com/python/time_clock.htm

[3] http://stackoverflow.com/questions/5152858/how-can-i-measure-a-duration-in-seconds-in-a-linux-shell-script