International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 10 Issue: 07 | July 2023 www.irjet.net p-ISSN: 2395-0072

Benchmarking Techniques for Performance Analysis of Operating Systems and Programs

Devansh Baid[1], Tanmay Mandal[2], Nilophar Joglekar[3], Akila Victor[4]

[123]Student, School of Computer Science and Engineering, VIT University, Tamil Nadu, India
[4]Associate Professor, School of Computer Science and Engineering, VIT University, Tamil Nadu, India

---***---

Abstract - *Benchmarking computer systems is an important, albeit time-consuming, process that provides insight into a system's performance, exposes weaknesses, and permits comparison between systems or versions. Current UNIX operating system benchmarking programs are heavily weighted toward CPU and hardware performance. As operating systems become more complicated, so do the programs that run on them. The commonly accessible benchmarks currently do not take this into account, making them kindly unrealistic and uninformative. We give a comparison of extensive and widely used operating systems. Eventually, a detailed and comprehensive conclusion has been reached based on the results of various tests.*

Key Words: Benchmarking; Operating Systems, Process Management, Task Scheduling, File Management, Computer Performance, Network Monitor

1. INTRODUCTION

In the early days of computing, the primary purpose of a programmer was to create a workable program with little regard for its efficiency. Von Neumann contrasted the speed with which early computers (including the ENIAC) conducted multiplication when computing ballistic trajectories in 1946 [McK88]. Herbst et al. measured the instruction mix of Maniac computer programs in 1955.

Performance evaluation is of importance to computer system designers, administrators, and users alike. Designers analyze several alternative designs to select the finest one. Administrators compare various systems to determine the optimal system for a collection of applications. Users examine many installed systems to determine which system is best suited for a certain task. The key objective in computer system design, procurement, and use is to provide maximum performance at the lowest possible cost.

The evaluation of a computer system's performance is a challenging task. When people hear the word "performance," they may conjure up entirely different images. persons who work with huge databases tend to think of performance in terms of transactions per second, but persons in scientific and technological fields may be interested in the number of floating-point operations per second. Even with a small emphasis, evaluating performance is not simple. Assume one is only interested in scientific computing. For scientific computing, a wide range of high-performance computers are available, ranging from vector computers with a small number of processors sharing common memory to machines with thousands of processors and distributed memories. The performance range of these computers can vary significantly, perhaps by a factor of a thousand or more, depending on how well the problem and software fit the underlying architecture and operating system.

2. RELATED WORK

There are numerous existing technologies for performance evaluation and benchmarking tools. Computer system designers, administrators, and users are interested in performance evaluation since their goal is to achieve or deliver the best performance at the lowest possible cost. As system performance varies greatly from one application area to the next, no one statistic can be used to assess computer system performance across all applications. Loads on various system components also have a significant impact on performance.

The performance of a computer system for a specific task can be measured using a variety of performance assessment and benchmarking tools. However, very few benchmarks, if any, examine how a system performs under various system loads. Most of the benchmarks are intended to run on an 'idle system'. As a result, they provide a measurement of a system's "peak efficiency" under a specific category of workload.

In the world of computing or computer networking, the benchmarking idea is not new. When one talks about "benchmarking tools," they typically mean a program or set of programs that are used to compare one solution's performance to that of another under specific reference circumstances. Benchmarking methods have been employed to rate the efficiency of computers and computer networks since the 1970s.

In the research projects CREW and OneLab2, Benchmarking of wired and wireless computer networks is presented as a study topic and the BonFIRE experiment investigates benchmarking of applications and virtual machines in an Infrastructure-as-a-Service (IaaS) setting.

They explained how the term "benchmarking" is used in these projects and addressed the question of why benchmarking research is still pertinent today. Following the presentation of a high-level generic benchmarking design, Benchmarking cloud services and cognitive radio systems are two instances that demonstrate the power of benchmarking.

3. PROBLEM STATEMENT

Performance evaluation is of importance to users, administrators, and designers of computer systems. The computer system testing program may behave differently depending on the input data and other settings in different runs of the application. Another issue is that the majority of installations are utilized to run different applications, and relatively few systems are specifically designed to perform one task. If the tasks that several apps undertake are of a similar character, test programs can be created to forecast how well the system will accomplish tasks of that nature.

4. AIM

To effectively measure performance, it is essential to identify the specific metrics to be evaluated in the design of a performance evaluation tool or benchmark. Historically, measuring CPU and disk I/O has been the norm. However, with the recent shift towards disk-less or data-less workstations, evaluating the performance of the file server and network has become critical, increasing the importance of measuring performance in these areas.

Typically, performance measurement tools are utilized to forecast the performance of an unfamiliar system on a well-defined or known task or set of tasks. The output of the performance evaluation is then utilized to make informed purchasing decisions when acquiring a new system. These tools can also serve as monitoring and diagnostic tools.

Running a test program and comparing the results against a known configuration can potentially identify the root cause of poor performance. Likewise, running a test program after making changes can determine whether there has been an improvement or degradation in performance.

Ideally, the most effective way to test a system's performance is to use the actual application that will run on the system. Unfortunately, this is not always feasible as the application may not be available before the system is purchased. Additionally, even if the application is available, the performance of the system can vary across different runs of the same application due to differences in input data and other variables.

A challenge arises from the fact that most systems are multi-purpose, supporting various applications instead of being dedicated to a specific task. If the applications perform similar job types, test programs can predict system performance for those types of jobs. However, educational institutions often use hybrid environments where the same resources are utilized for various applications unless an application is highly specialized.

The performance of various subsystems (such as the CPU, disc I/O, etc.) is measured, and the results are given as a set of numbers, rather than a single number, as a result.

5. LITERATURE REVIEW

Several researchers, vendors, and organizations have conducted theoretical and experimental performance analyses and studies of various operating systems in recent years. Professor B. Randell presented the intricate difficulties in achieving satisfactory levels of performance in operating systems in his paper (Randall, 1985), emphasizing the strong relationship and trade-off with reliability. His research focused on the two issues (performance and reliability) that designers and implementers face.

Another study conducted by a group of Harvard University researchers (Ahmed, 2010) compared the performance of Windows for Workgroups, Windows NT, and NetBSD (a UNIX variant OS) across a wide range of system functionality and user requirements. They tracked the performance differences between the systems and attributed them to the kernel architecture. Windows employs a microkernel architecture, whereas NetBSD employs a monolithic kernel structure. They demonstrated that Windows generated more overhead when running native applications due to frequent changes in machine mode and the use of system call hooks. In another dimension of performance measurement, Saavedra et al. (1996) in their work used a machine-independent mathematical model to characterize machine performance and program execution (as opposed to measuring experimentally). They used this model to estimate execution time for arbitrary machine/program combinations.

In this abstract operation, benchmarks called and programs were used to measure the execution of a given operation, and this was used to predict the execution time on a large spectrum of machines. Although this work is almost 20 years old, it offers up some interesting observations and contributes to the field of benchmarking by furthering the research and discussion to include some answers to the and behind the better performance or lack thereof, and in addition, explores the act of predicting execution performance.

In the experiments associated with this research paper, Java I/O-intensive, and CPU-intensive benchmarks were designed and written, and executed concurrently (in parallel) and their execution times were measured for different sets of input data.

The benchmarks were run via Linux scripts and Windows batch files via the command prompt and terminal for the respective platforms under study.

6. PROPOSED SYSTEM

A. System Design

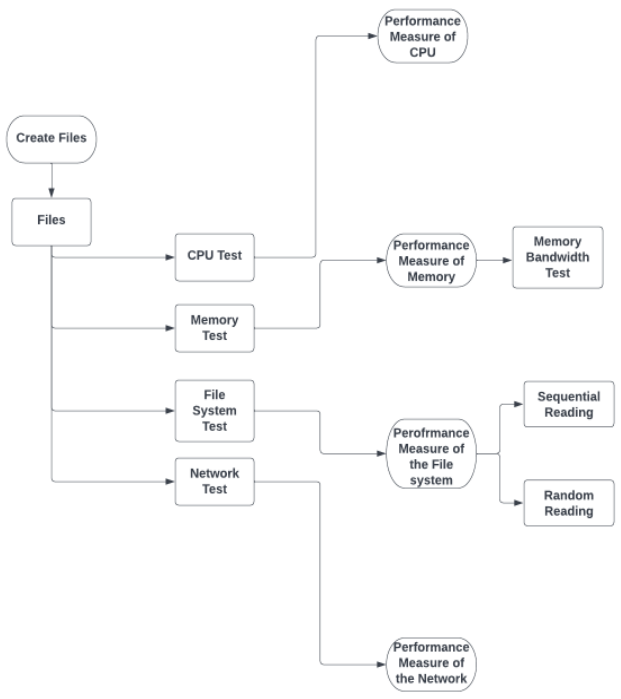

Create Files
Files
CPU Test
Memory Test
File System Test
Network Test
Performance Measure of CPU
Performance Measure of Memory
Memory Bandwidth Test
Perofrmance Measure of the File system
Sequential Reading
Random Reading
Performance Measure of the Network

Fig -1: Block Diagram

Dummy Files are created to test the performance of our system on various parameters

- Performance measure of CPU - We measure the number of instructions executed by the CPU system in a given clock cycle.
- Performance measure of memory - We measure the performance of the memory by conducting a regular memory bandwidth test.
- Performance measure of the file system - We measure the performance of the file system by conducting two tests-
    1. Random reading
    2. Sequential reading
- Performance measure of the network - We measure the performance of the network by conducting network tests.

B. Benchmarking System Design

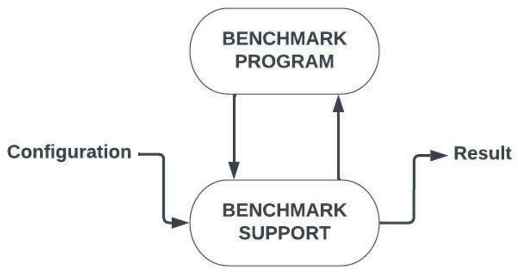


Fig -2: Benchmarking System Design

Modifying our benchmarking software so that it can operate on various operating systems, where it will be put through a variety of performance tests to verify our system. After receiving the benchmark findings, we will compare them to accepted system metrics.

7. METHODOLOGY

A. CPU Benchmarking

- We chose a benchmarking method that can quickly access memory.
- Wrote a C program that performs the mentioned benchmarking method. This program should run the benchmark repeatedly for a specified amount of time and display the results.
- Analyze and display the results through visualization techniques that could enable determining CPU's performance with ease.

B. Memory Benchmarking

- Allocate a memory block larger than the cache size of the system.
- Initialize the memory block with a pattern to ensure full population.
- Perform a read or write operation on the entire memory block while measuring time.
- Calculate memory bandwidth and display results.

C. Filesystem Benchmarking

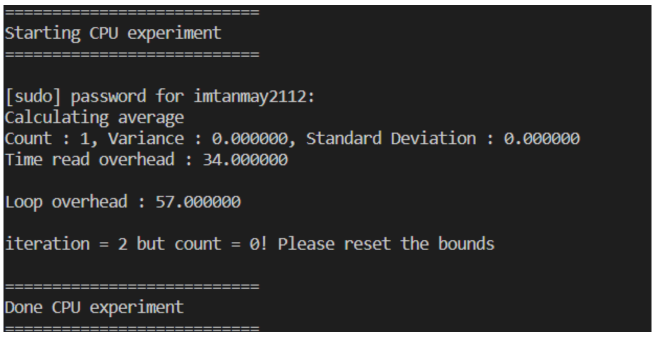
- Choose a file operation and create a file larger than the file system's block size.
- Initialize the file with some data and start a timer.
- Perform the file operation and stop the timer.
- Calculate the file system performance and display the results in a meaningful way.

D. Network Test

- Choose a network protocol and create a socket bound to a local IP address and port number.
- Initialize the data to be sent and start a timer.
- Send the data over the network and receive it using the same socket, then stop the timer.

- Calculate the network performance and display the results in a meaningful way.

8. IMPLEMENTATION



```
==========================
Starting CPU experiment
==========================

[sudo] password for imtanmay2112:
Calculating average
Count : 1, Variance : 0.000000, Standard Deviation : 0.000000
Time read overhead : 34.000000

Loop overhead : 57.000000

iteration = 2 but count = 0! Please reset the bounds

==========================
Done CPU experiment
==========================
```

Fig -3: Outcome of CPU Test

By using multiple processes, we are calculating the time taken by the CPU to execute an instruction cycle.

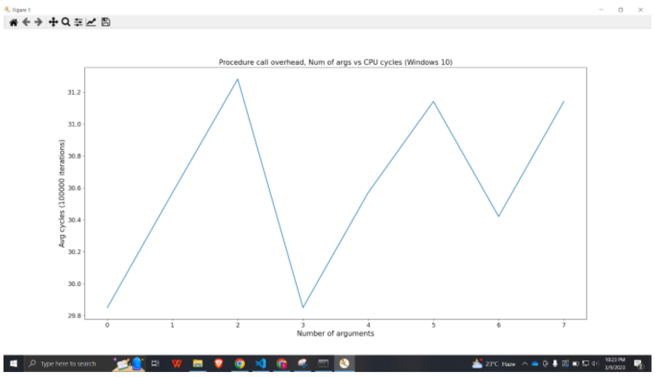
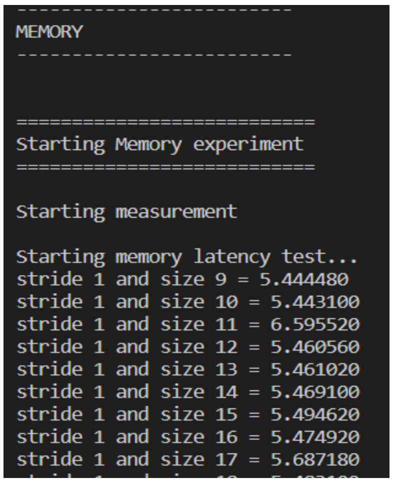


Fig -4: Plot for Average Cycles vs Number of Arguments

This plot depicts gives a better understanding on how much time our CPU is taking to execute an instruction cycle.



```
-------------------------
MEMORY
-------------------------


============================
Starting Memory experiment
============================

Starting measurement

Starting memory latency test...
stride 1 and size 9 = 5.444480
stride 1 and size 10 = 5.443100
stride 1 and size 11 = 6.595520
stride 1 and size 12 = 5.460560
stride 1 and size 13 = 5.461020
stride 1 and size 14 = 5.469100
stride 1 and size 15 = 5.494620
stride 1 and size 16 = 5.474920
stride 1 and size 17 = 5.687180
```

Fig -5: Outcome of Memory Latency Test

We are using the concepts of strides with different sizes to go through the data and then calculating the latency to fetch the data.

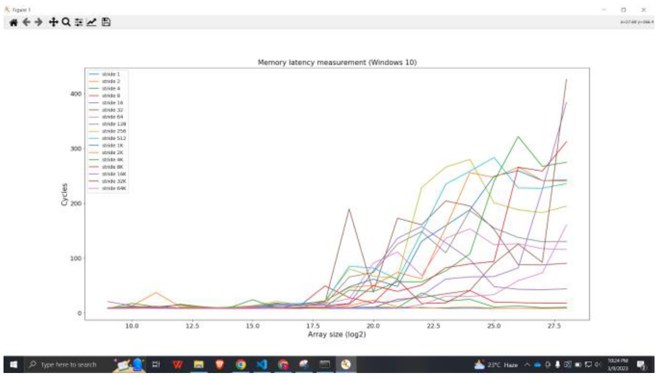
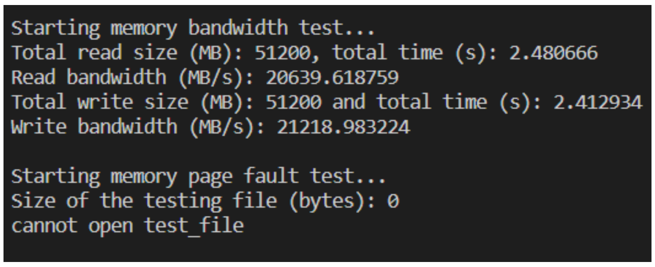


Fig -6: Plot for Memory Latency w.r.t different strides

This plot gives a clear description of the memory latency test in accordance with the strides we have used.

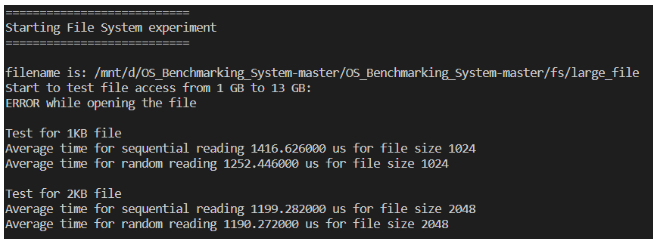


```
Starting memory bandwidth test...
Total read size (MB): 51200, total time (s): 2.480666
Read bandwidth (MB/s): 20639.618759
Total write size (MB): 51200 and total time (s): 2.412934
Write bandwidth (MB/s): 21218.983224

Starting memory page fault test...
Size of the testing file (bytes): 0
cannot open test_file
```

Fig -7: Outcome of Memory Bandwidth Test

Here, we calculate the time taken to read the data by using timestamps. These timestamp values are captured before the process of reading and writing.



```
=========================
Starting File System experiment
=========================

filename is: /mnt/d/OS_Benchmarking_System-master/OS_Benchmarking_System-master/fs/large_file
Start to test file access from 1 GB to 13 GB:
ERROR while opening the file

Test for 1KB file
Average time for sequential reading 1416.626000 us for file size 1024
Average time for random reading 1252.446000 us for file size 1024

Test for 2KB file
Average time for sequential reading 1199.282000 us for file size 2048
Average time for random reading 1190.272000 us for file size 2048
```

Fig 8: Outcome of File System Experiment

For testing the file system, we used dummy test files which were of different sizes. We then conducted the sequential reading as well as random reading tests with the assistance of timestamps.

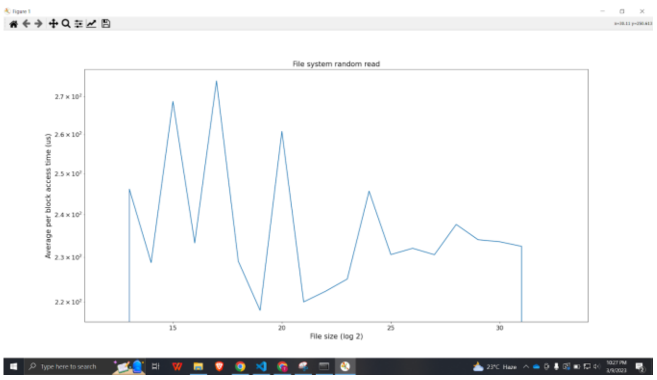


Fig -9: Plot for the average per block access time for different file sizes during Random Read Test

This plot clearly depicts the time taken for the random read test with respect to the size of the file.

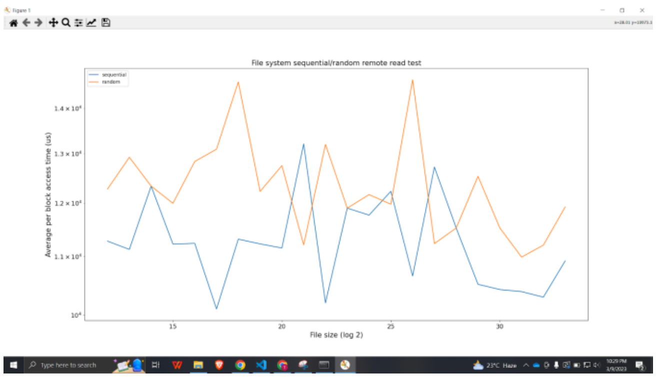
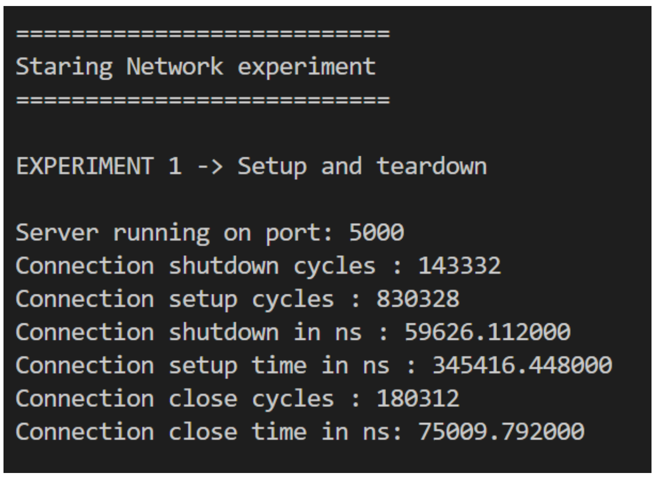


Fig -10: Plot for comparing File System Sequential Read Test & Random Read Test

This plot clearly shows the comparison between the file system test and the random read test done by the system.



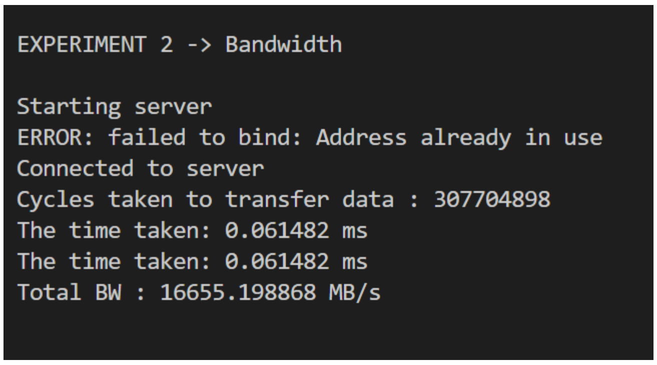
```
============================
Staring Network experiment
============================

EXPERIMENT 1 -> Setup and teardown

Server running on port: 5000
Connection shutdown cycles : 143332
Connection setup cycles : 830328
Connection shutdown in ns : 59626.112000
Connection setup time in ns : 345416.448000
Connection close cycles : 180312
Connection close time in ns: 75009.792000
```

Fig -11: First experiment for network test

We are using the concepts of sockets to visualize the network performance of the system. The first experiment is the time taken to setup the connection.

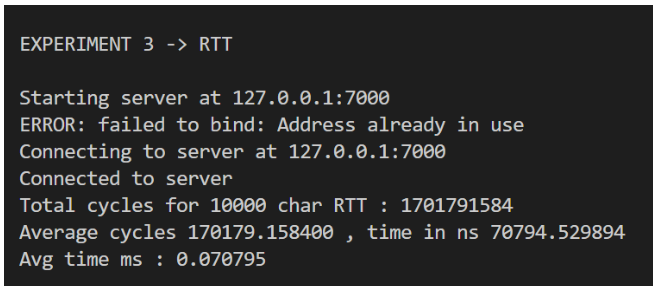


```
EXPERIMENT 2 -> Bandwidth

Starting server
ERROR: failed to bind: Address already in use
Connected to server
Cycles taken to transfer data : 307704898
The time taken: 0.061482 ms
The time taken: 0.061482 ms
Total BW : 16655.198868 MB/s
```

Fig -12: Second experiment for network test

The second experiment is the bandwidth test, in this test, we calculate the time taken to transfer data.



```
EXPERIMENT 3 -> RTT

Starting server at 127.0.0.1:7000
ERROR: failed to bind: Address already in use
Connecting to server at 127.0.0.1:7000
Connected to server
Total cycles for 10000 char RTT : 1701791584
Average cycles 170179.158400 , time in ns 70794.529894
Avg time ms : 0.070795
```

Fig -13: Third experiment for network test

In this experiment, we calculate the RTT (Round trip time). It is the time taken for a network request to go from a starting point and back again to the starting time.

9. CONCLUSIONS

We have successfully developed a benchmarking tool that has the capability to test the system, irrespective of the OS the user is using. Since our tool focuses on testing the system on different aspects individually, it can assist the user by saving him some time by not looking at the system holistically and providing him with an analysis of where an issue is occurring. Moreover, all the testing is done locally on the system so it showcases the performance of the system realistically. Future work for this tool would be to enhance performance metrics, develop a user-friendly interface and integrate some cloud-based solutions.

REFERENCES

[1] Vetrivel, P., Sivakumar, S., & Babu, K. S. S. A Survey of Benchmarking Techniques for Real-Time Operating System Performance Analysis.

[2] Maly, K. J., Gupta, A., & Mynam, S. K. (1996). A workstation's communication performance benchmark. In Testing of Communicating Systems (pp. 61-76). Springer, Boston, MA.

[3] Boras, M., Balen, J., & Vdovjak, K. (2020, October). Performance Evaluation of Linux Operating Systems. In 2020 International Conference on Smart Systems and Technologies (SST) (pp. 115-120). IEEE.

[4] Martinovic, G., Balen, J., & Cukic, B. (2012). Performance Evaluation of Recent Windows Operating Systems. J. Univers. Comput. Sci., 18(2), 218-263.

[5] Vdovjak, K., Balen, J., & Nenadić, K. (2020). Experimental Evaluation of Desktop Operating Systems Networking Performance. International journal of electrical and computer engineering systems, 11(2), 67-76.

[6] Shreesha Rao P, Chandan K N, 2014, The Need For Portable Benchmark to Evaluate The Performance of Real Time Operating Systems, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) NCRTS – 2014 (Volume 2 – Issue 13)

[7] Kalakech, A., Jarboui, T., Arlat, J., Crouzet, Y., & Kanoun, K. (2004, March). Benchmarking operating system dependability: Windows 2000 as a case study. In 10th IEEE Pacific Rim International Symposium on Dependable Computing, 2004. Proceedings. (pp. 261-270). IEEE.

[8] Waller, J., & Hasselbring, W. (2013). A benchmark engineering methodology to measure the overhead of application-level monitoring. CEUR Workshop Proceedings.

[9] Chen, J. B., Endo, Y., Chan, K., Mazieres, D., Dias, A., Seltzer, M., & Smith, M. D. (1995). The measured performance of personal computer operating systems. ACM SIGOPS Operating Systems Review, 29(5), 299-313.

[10] Marieska, M. D., Hariyanto, P. G., Fauzan, M. F., Kistijantoro, A. I., & Manaf, A. (2011, December). On performance of kernel based and embedded real-time operating systems: Benchmarking and analysis. In 2011 International Conference on Advanced Computer Science and Information Systems (pp. 401-406). IEEE.

[11] Maly, K., Gupta, A., Mynam, S., & Khanna, S. (1995). B (its) T (o the) U (ser): A Communication Benchmark Proposal. Technical Report. Department of Computer Science, ODU.

[12] Kanoun, K., Crouzet, Y., Kalakech, A., & Rugina, A. E. (2008). Windows and linux robustness benchmarks with respect to application erroneous behavior. Dependability Benchmarking for Computer Sys, 227-254.

[13] Jeong, T. K. (2006). Evaluation and Benchmarking on Operating System for Embedded Devices. Journal of the Korea Institute of Information and Communication Engineering, 10(1), 156-163.

[14] Hong, P., Hong, S. W., Roh, J. J., & Park, K. (2012). Evolving benchmarking practices: a review for research perspectives. Benchmarking: An International Journal.

[15] Kanoun, Karama & Crouzet, Yves Y.. (2006). Dependability Benchmarks for Operating Systems. International Journal of Performability Engineering. 2. 277-289.

[16] Gomez, J., Tran, T. V., & Sadre, R. " Benchmarking framework for real-time operating system applications: study and implementation with Contiki and RIOT.

[17] Koopman, P., Sung, J., Dingman, C., Siewiorek, D., & Marz, T. (1997, October). Comparing operating systems using robustness benchmarks. In Proceedings of SRDS'97: 16th IEEE Symposium on Reliable Distributed Systems (pp. 72-79). IEEE.

[18] Sim, S. E., Easterbrook, S., & Holt, R. C. (2003, May). Using benchmarking to advance research: A challenge to software engineering. In the 25th International Conference on Software Engineering, 2003. Proceedings. (pp. 74-83). IEEE.

[19] Kalakech, A., Kanoun, K., Crouzet, Y., & Arlat, J. (2004, June). Benchmarking the Dependability of Windows NT4, 2000 and XP. In International Conference on Dependable Systems and Networks, 2004 (pp. 681-686). IEEE.

[20] Narayan, S., Shang, P., & Fan, N. (2009, April). Performance evaluation of ipv4 and ipv6 on windows vista and linux ubuntu. In 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing (Vol. 1, pp. 653-656). IEEE.