임베디드 디바이스를 위한 운영체제의 벤치마킹과 성능평가

정 태 경*

Evaluation and Benchmarking on Operating System for Embedded Devices

Taikyeong Jeong*

요 약

본 고에서는 임베디드 시스템을 위한 운영체제를 평가하고 성능을 검사하는 것을 주안점으로 삼고 있으며 현재 상용화 되어있는 윈도우즈 플래포움을 기초로 하고 있다. 전형적인 컴퓨터의 workload 이용하고 시스템성능을 증가시키는 운영체제의 벤치마킹의 분석을 포함한 기본적인 방법과 동시에 하위레벨에서의 규명을 이루 고 있다. 본 고에서는 "WinStone" 과 "HBench" 같은 선택되어진 어플리케이션중심과 직접적으로 시스템에 영향을 끼치는 가장 최선의 벤치마킹툴이 사용되었다. 이 실험과 케이스 연구를 통하여 벤치마킹툴을 이용하여 윈도우즈 플래포움의 하위레벨 테스트와 동시에 어플리케이션 레벨의 임베디드 마이크로시스템의 성능을 보여 주고 있다.

ABSTRACT

The objective of this paper is to evaluate the performance of an operating system for embedded devices to that of the commercially available Windows platform. Analysis includes benchmarking the operating systems using a "typical" PC workload, as well as identifying low-level areas in which the updated OS is limiting or enhancing the system performance. The primary benchmarking suites selected for this paper are "WinStone" and "HBench", with the former providing an application-based suite of tests and the latter providing the most direct means for isolating operating system effects on the system. We have demonstrated in a case study for embedded microprocessors, and evaluated a Windows platform at a low-level test as well as an application level using a benchmarking suite.

키워드

Operating System, Embedded System, Computer Architecture, Benchmarking, Workload

Ⅰ. INTRODUCTION

As embedded systems, such as PDAs, mobile phones, point-of-sale devices, VCRs, industrial robot control, or even toasters, become more complex hardware-wise with every generation, applications they run require more and more actual operating system code in order to keep the development time reasonable

Also, an operating system for embedded systems is arguably the most important component that go into a computer [1]. The purpose of an operating system is to serve as an interface between the user and/or application programs

* T. Jeong was with Department of electrical and computer engineering, the University of Texas at Austin, Austin, TX 78712. Email: ttjeong@alumni.utexas.net, Tel: +1-512-786-6402)

접수일자 : 2005. 10. 21

and the computer hardware. The operating system is responsible for exercising overall control of the computer including allocating of resources, scheduling of operation, and enforcing security and protection between separate programs, processes and/or users [2]. The main purpose of the operating system is to maximize the utilization of the computer's resources while providing users with the perception that they have sole ownership of those resources

Since the early 1990s, the PC platform has been largely dominated by Microsoft's Windows Operating System. In today's world of media frenzy and marketing, it is rare that products are truly compared on the basis of their merits.

Consumers must critically evaluate the manufacturer's benchmarks to screen against misleading metrics and to ensure benchmarks accurately reflect the intended workload of the systems. These study first attempts to determine what differences, if any, exist between the embedded operating systems at a low, primitive, functional level and then compares the performance of each operating system for embedded devices under an application-based workload.

In this paper, we observed an operating system that suits the needs of a general purpose computer as well an embedded processor. Windows is designed to provide manageability, reliability and flexibility for many types of applications in the embedded system area. As with any operating system, Windows platform must efficiently handle user input and system resource utilization.

The remainder of the paper is organized as follows. In Section 2, we describe related evaluation methods. In Section 3, we address the case study in the context of embedded systems. Section 4 presents the details of the test environment. In Section 5, we report the result of our test for evaluating the operating system performance. Finally, Section 6 concludes the paper with a discussion of future work.

II. EVALUATION OF OPERATING SYSTEM

In order for an operating system to be effective, it must demonstrate some key characteristics. These include concurrency, sharing, long-term storage, and non-determinacy [3]. Concurrency deals with the existence and ability to handle several simultaneous processes with the aim of servicing each such that the existence of other active processes is not perceived. Concurrent processes will likely have to share the limited computer resources, programs and/or common data. The operating system must provide for efficient distribution among active processes.

As processes share resources and programs, the need for internal long-term storage arises, Internal storage also provides a convenience for the user. Although process should produce the same results each time they are run, the operating system must be able to be run in an unknown and variable internal environment. The outcome of a process must not be affected by the operating system's responding to numerous contingencies (i.e., other processes, interrupts, errors, etc).

In addition to the above requisite characteristics, there are several desirable features operating systems should display. Although not functionally essential, given increasing computer complexity and market competition, most operating systems today display all of the following features: easy to use interface, efficiency, flexibility, security/integrity, reliability, maintainability, and transparency to the user [4].

A. Functionality of Operating System

The function of an operating system can be grouped into two components: management and support. The management component controls the computer and is composed of three sub-functions: (i) job management, (ii) task management, and (iii) data management [2]. Job management deals with program evaluation and execution, while task management deals with lower level component tasking.

A process is a system activity such as a program in execution, a time-shared user program, or a subsystem task [4]. Once a process is initiated, either as the result of a parent's fork()operation or internal system generation, it is the operating system that evaluates the requirements of that process and controls its execution.

The data management component embodies several functions to include memory, secondary storage, input/output (I/O), and file management [4]. Broadly stated, the goal of

data management is to provide an efficient, logical way to store and transfer data while "hiding"hardware peculiarities from the user. Data management mechanisms must be independent of both the physical form with which the date is stored as well as information represented by the data.

B. Support Component for System Function

The second component of an operating system, the support component, consists of functions that support the management component as well as provide aids for the user. Although many of the support functions seem similar to some of the management functions, they are distinguished by their functionality. The support component can be divided into three general sub-categories: (i) service functions, (ii) system calls, and (iii) system programs. [4].

System calls provide the interface between a running program and the operating system. Programs use system calls to cause certain operating system functions to be performed during execution. Programs can use system calls to perform any of the following functions: process and job controls, file manipulation, device management, and information management [4].

System programs solve common problems and provide a more convenient environment for program development and execution. If system programs were not included as part of the operating system, it is likely that the user would want to write their own programs to perform similar operations. The operating system provides system programs to facilitate file manipulation, status information, file modification, program loading and execution, and common application programs [4].

Ⅲ. A CASE STUDY

The motivation for researching this case is fairly straightforward. First, it is interesting to determine if the type Microsoft had created about NT5.0 platform had any real foundation. In addition, they have been touting this new version as the "premier" operating system of the future, claiming that it will serve everyone's needs in the best way possible.

A. An Analysis on Windows NT platform

The Windows NT platform is a powerful operating system that suits the needs of network administrators as well as the normal user. Windows NT platform is designed to provide manageability, reliability, and flexibility for many types of applications. As with any operating system, the Windows NT platform must efficiently handle user input and system resource utilization.

Yet, another reason to investigate this operating system was because Windows-based systems are by far the most widespread and their presence is growing daily. Plus, the new version has an impressive list of enhancements. It will include a completely revamped file system that will make it more "Internet friendly"and easier to view multiple drive volumes on a network [8]. Security will become even greater with file-level encryption. It has been designed to look and feel more like its easy-to use cousin Windows 95/98, including a high level of Internet integration. Configuration will be much easier with its support for Plug and Play devices, as well as other peripheral (DVD, USB,and Firewire). Probability features will include both partial shutdown as well as hibernation for resuming without actually shutting down. This is only a partial list of the innovations to be included with the NT5.0 platform.

B. Architecture and File System

Much of Windows NT's renowned stability is a direct result of the operating system's architecture that uses "modified microkernel" architecture. Core operating system services used for thread scheduling multiprocessor synchronization, and other low-level tasks are implemented in the microkernel. A separate layer on top of the microkernel controls virtual memory management, process management, and other operating system services. The microkernel itself is insulated from the physical characteristics of timers, interrupt controllers, and other hardware devices by services in the Hardware Abstraction Layer (HAL). These control features are referred to as "Self-Organizing Computer System [6]".

Modern embedded microprocessors allow paths taken

through code stored in memory (threads of execution) to run at various privilege levels. Code running at a low privilege level can't access data in memory marked with higher privilege levels. The NT Executive runs in the highest privilege level, the kernel mode. Intel x86 processors supports four privilege levels, refereed to as rings 0 to 3, and kernel mode is synonymous with ring 0. Applications are run in user mode(ring 3 on Intel CPUs). This makes it impossible for an application to corrupt code or data stored in the Windows NT Executive because the processor will notify the operating system of attempted invalid memory accesses [6].

One notable feature found in the Windows NT's architecture is that portions of the operating system, referred to as the subsystems, run user-mode processes alongside application processes. The subsystem provides the environments in which applications run. The Win32 subsystem provides the application program interface (API) services that Windows application calls to do such things as creating windows and opening files. When a Windows application calls an API function in the Win32 subsystem, which may happen several hundred times per second, the operating system's original API services out of the Win32 subsystem and into the operating systems kernel to yield a performance boost and reduce memory requirements. Moving the windows manager to the kernel eliminated a key bottleneck, numerous context switches, which limited bandwidth between the window manger and application programs that call window manger services [6].

Windows NT4.0 offers users multiple File System methods. Originally, with operating systems such as DOS, and continuing through the Windows 95/98 line, users would format their drives into FAT partition. Windows NT4.0 offers NTFS (NT File System) formatting as well. One of the main advantages of an NTFS partition over a FAT partition is that an NTFS disk partition can't become fragmented. The performance of a FAT partition can become considerably deteriorated due to fragmentation, which is why there are tools to repack, or defragment, a FAT partition. Microsoft maintains that NTFS doesn't need a defragment tool, that it is effectively self-defragmenting. The "large fluffy disk"heritage of the FAT format is gone, replaced by a properly designed and structured file system that has some similarities to Unix file system. Microsoft has barely documented many of the NTFS performance features, although most are considered "standard accepted practice" for sensible disk systems [7].

With Windows NT 5.0, the FAT32 file system was introduced. FAT32 offers an alternative to the inflexible FAT system without having to incur the overhead associated with the NTFS system. FAT32 mangers data in a similar manner to the FAT system but provides the user some flexibility in establishing file system parameters. FAT32 permits partition sizes in excess of the 2GB maximum that was imposed under FAT and also permits variable cluster sizes so the user may "custom design" the file management parameters to best suit the needs of the workload.

IV. TEST ENVIRONMENTS

In this section, we describe a test environment of operating system that will be considered for the embedded microprocessor. The test environment is a Dell precision 410 workstation. It utilizes a Pentium II processor with a clock speed of 400 MHz, 256 MB of main memory, 32 KB L1 cache, 512KB L2 cache, 4GB EIDE hard drive, and SCSI CD-ROM.

A. Benchmarking Tools

As mentioned earlier, to obtain the most complete evaluation of the Windows 5.0 operating system for embedded devices, we plan to conduct two different types of tests. The first using HBench-OS is designed to evaluate the operating system at a low level. The second using WinStone 98 suite is designed to evaluate the operating system under a realistic, application based workload.

B. HBench-OS Suites

This evaluation involved determining how well the operating system was interacting with the hardware, including the processor, BIOS, video subsystem, hard drives, and any other hardware subsystem. To do this we used a tool

developed by Aaron Brown and Margo Seltzer at Harvard University [9] called HBench-OS. HBench is suite of benchmarks designed to measure the performance of primitive functionality provided by an OS/hardware platform. The primary reason for selecting this tool is the simple fact that it was the only low-level benchmark readily available that we could find. HBench runs tests to determine:

- Memory bandwidth and latency
- File transfer bandwidth and latency
- Process communication bandwidth
- Context switch latency
- File system management latency
- System call latencies
- Networking bandwidth and latency

One problem we encountered with HBench is the fact that it was written for a Unix environment. Since Windows NT is far form being similar to a Unix-like environment, we had two choices. We could rewrite the entire test to work under NT, or we could find some way to emulate Unix in the NT environment.

We used the file system type and cluster size as the variables for our experiments. We ran the HBench test on NTFS, FAT16 and FAT32 with varied cluster sizes for both NT4.0 and NT5.0. The tables below shows the exact tests run:

Table 1. HBench Test Parameters

Size of Clusters	NT5.0			NT4.0	
	NTFS	FAT32	FAT16	NTFS	FAT16
512	x	x		x	
1024	x	x		x	
2048	x	x	x	x	x
4096	x	x	x	x	x
8192			x		x
16384			x		x

Each test run included six iteration of each of the HEench tests of which the middle four results were used to provide an average for that test. To insure fairness the system was reset to the same state between each test run. A new partition was created for each new test run, and the remaining partitions were defragmented. Also, no user programs were running during the tests.

C. WinStone 98 Suites

The second type of tests is a more "user" level test. Seeing how the system would perform from the user's eyes is the primary motivation behind this type of testing. There are several common benchmarking suites that perform such test. One of the most popular is ZiffDavis's WinStone, which does an application-based evaluation. WinStone opens several of the most popular software packages, such as Microsoft Office and Internet Explorer, and automatically performs typical user actions, including typing, file saves, mouse clicks, cut and paste, etc. The WinStone suite presents the most "lifelike" workload to the system possible, while maintaining continuity across separate test runs.

We ran the entire suite of test on both Windows NT4.0 and NT5.0. Because HBench should reveal any performance differences due to file system and cluster size, we did not vary those parameters for the WinStone tests. To obtain an "out-of-the-box" performance estimate, the tests were run using the default NT drive parameters, NTFS with a 1KB cluster size. Again, no user programs were open during the testing.

V. RESULTS AND ANALYSIS

In this section, we discuss two test results, HBench at a low-level and WinStone at an application level, for evaluating the operating system performance.

A. Hbench-OS Suite (Primitive Functionality)

In particular, the networking tests were not used because our testing environment did not support networking. In addition, several of the tests would not compile properly or were deemed nonessential given the suite of test that would compile properly. A description for each of the tests that were run and a portion of the results from each is given below.

A.1. Memory Copy Bandwidth

The memory copy bandwidth test determined the

maximum throughput possible between two buffers in memory. It could use either the operating system's libc bcopy() or memcpy() or an unrolled loop. It took three parameters - the size of the memory buffer to copy, whether to use the OS's libc copy routine or an unrolled loop and whether to use buffers that are page-aligned or displaced relative to each other. The following results are for an NTFS file system using memory and an unaligned buffer.

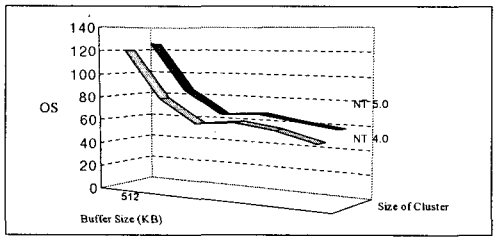

Fig 1. Memory Copy Bandwidth Comparison

Figure 1 shows comparison result of memory copy bandwidth. There is little difference in the memory performance between the two versions of NT. This parallelism holds regardless of the file system, thus the FAT results are not included. Coincident with intuition the results for this test were independent of the cluster sizes and file system types. This was expected since virtual paging should not be necessary due to the size of our RAM, the hard drive should not be accessed during this test. One noticeable characteristic of these graphs is the stair-stepping they seem to exhibit. As the buffer size is increased the bandwidth not only decrease, but it does soin seemingly quantized units. It is hard to know exactly why this occurs. Most likely this relates to the cache and its effects on memory operations. Since caching is done in blocks and the steps seems to coincide with the system level cache size (L1 and L2 cache), each "step-down" probably indicates an overflow to another block in cache memory. For this project, the most important observation to be made from these graphs is the simple fact that NT5.0 and NT4.0 performed at about the same level in all cases.

A.2. Basic Context Switch Latency

Context switching is a very important feature of most operating systems today. This procedure involves saving the current state of the machine for the running process, selecting the next process to be run and resorting the machine to the state of the new process. A process state includes the value of the instruction counter, all of the data registers, among other things. If this switching occurs too often, the operating system will incur too much overhead processing, having a significant effect on the system's performance.

To test the latency of a context switch, HBench passes a token through a series of pipes connecting different processes. The HBench authors defined the context switch latency to be "the time it takes the systems to switch hardware contexts and to update hardware TLBs, etc [9]. They explain further that they "specifically exclude any latency due to satisfying cache conflict misses that are artifacts of the context switch" because they do not feel that this is overhead that the operating system forces upon the system. The parameters for this test include the size of the memory footprint of the test processes as well as the number of processes to be run simultaneously. Our parameters included varying the memory footprint from 0-64 KB and varying the number of processes from 2-20.

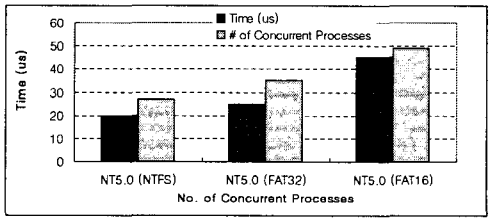

Fig 2. Context Switching Latency
(with 8K footprint, 2KB Cluster size)

Figure 2 illustrates results for this test under Windows NT5.0 platform. Unfortunately, this test would not run successfully under NT4.0, causing the system to hang when it was executed. Some interesting observations can be made about NT5.0, however. As is evident, the results were essentially identical for the different types of file systems. This makes perfect sense since the file system should not be involved in these operations. Nonetheless, the latency does increase as the number of processes is increase. Of special interest is the fact that the latency increases quite rapidly

when changing from one or two processes up to around five processes. The change in latency when above five processes is slight in comparison. NT5.0 no doubt uses several data structures to store the information for all processes that are currently running on in the queue. As the number of processes increases the amount of overhead necessary to manipulate all of these data structures increases dramatically as well. The initial increase in overhead for handling this data explains the initial jump in latency. Any additional processes still increase the amount of overhead, but not as much as at first.

A.3. Latency of the File System Layer (VFS layer)

Yet another aspect of the file system is being tested. The purpose of this test is to determine the basic overhead of the operating system's functionality by writing one byte of data to/dev/null/.

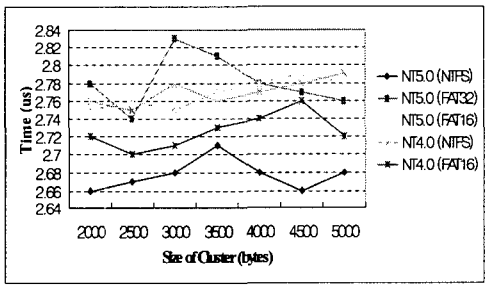

Fig 3. File System Layer Latency

The results for this test are quite conclusive. The upgrade has finally outperformed its predecessor!. NT5.0 has a much better implementation of its file system handling mechanism than does NT4.0. This is noteworthy because NT5.0 achieved better performance despite the increased functionality of its file management system such as improved security and scalability. Although the details are unknown, it is obvious Microsoft has incorporated some form of optimization to offer the additional overhead associated with NT5.0. Notice there was little variation between the different file system types for a given version of NT. This implies that the HBench test did a good job of isolating the operating system's overhead form the file system's overhead.

B. WinStone Suite (Application Level)

The WinStone benchmarking tool opens a number of the most popular software packages and emulates a typical use workload. Some of the programs it uses are Microsoft Access, Microsoft Excel, Microsoft Work, Corel Draw, Adobe PhotoShop, Microsoft Internet Explorer, and many more. It not only emulates typing, but mouse clicks, drag and drop, cut and paste, and many of the most popular functions. WinStone returns a single numeric value, in its own WinStone units, for each types of test that was run. This makes interpreting the results quite straightforward. The graph below shows the average results specific tests from the WinStone suite.

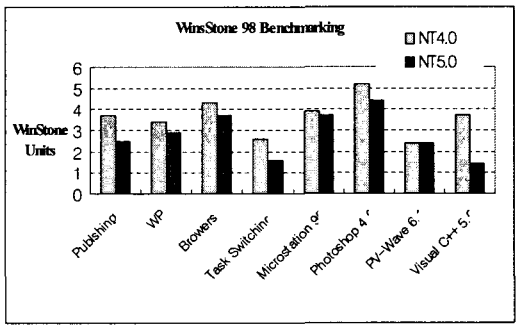

Fig 4. Results of WinStone 98 Benchmarking Suite

The surprising conclusion, obvious from the results, is that Windows NT4.0 performs better thanNT5.0 for each application. This result is counterintuitive, as reason dictates that for an upgraded version to be successful, it should outperform its successor.

The more robust user interface, increased data security, and other features included in Windows NT5.0 undoubtedly add overhead and this is reflected in a performance decrease from that of NT4.0 at the application level.

Ⅳ. CONCLUSION

A number of systems are marketed as embedded operating systems, often with the additional adjective "embedded devices." Rather than presenting a description of operating systems generally, this paper addresses knowledge of operating system concepts for embedded microprocessors in

pointing out an usage of operating system for embedded devices. We concluded with more or less "traditional" operating system issues, reconsidered in the light of embedded operating systems.

It is likely that the increased functionality associated with Windows platform is the cause of this decreased system usages. Designers of embedded devices face more constraints than designers of general-purpose devices. Power, energy, efficiency, cost, and physical dimensions usually have a much bigger role in embedded systems. Further research to pinpoint the exact cause(s) of Windows's reduction in application-level performance requires different benchmarking techniques than included in this study and warrants additional research.

참고문헌

[1] A. M. Lister, "Fundamental of Operating Systems", 3rd Edition, Springer-Verlag New York Inc., New York, 1984

[2] D Patterson and J. Hennessey, "Computer Architecture and Organization", McGraw-Hill Book Company, New York, 1988

[3] S. A Kurzban, T. S. Heines and A. P. Sayes, "Operating Systems Principles", 2nd Edition, Van Nostrand Reinhold Company, New York, 1984

[4] A. Silberchatz and J. Peterson, "Operating System Concepts", Addition-Wesley Publishing Company, 1989

[5] L. Cai, H. Yu and D. Gajski, "A Novel Memory Size Model for Variable-Mapping in System Level Design," Asia and South Pacific Design Automation Conference (ASP-DAC 2004), Yokohama, Japan, pp 813-818, January 27-30, 2004

[6] M. Bennani and H. Ruan, and D. Menasce, "On the Use of Online Analytic Performance Models in Self-Managing and Self-Organizing Computer Systems", Lecture Notes in Computer Science, Vol. 3460, Springer Verlag, 2005.

[7] O. Sinanoglu and A. Orailoglu, "Compacting Test Responses for Deeply Embedded SoC Cores," IEEE Design & Test of Computers, pp 22-30, July-August 2003,

[8] S. Banerjee and N. Dutt, "Efficient Search Space Exploration for HW-SW Partitioning", International Conference on Hardware/Software Codesign and System Synthesis, 2004, Stockholm, Sweden, pp 122-127, September 8-10, 2004

[9] HBench-OS Operating System Benchmarks, http://eecs.harvard.edu/~vino/perf/hbnech/

[10] J. A. Fisher, P. Farabosch and C. Young, Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools, San Francisco, CA, Morgan Kaufmann Publishers (an imprint of Elsevier), 2005

저자소개

Taikyeong Jeong received the Ph.D. degree from the Department of Electrical and Computer Engineering, the University of Texas at Austin in 2004. He performed research in the area of high performance circuit design and power efficiency system design. He joined the University of Delaware, where he is now a research associate under the research grants of NASA (Grant No. NNG05GJ38G) in 2004, working on high performance VLSI design for next generation space robotics devices and system. His research interests include VLSI design, computer architecture, embedded system, and high performance system design.