Advanced Non-Distributed Operating Systems Course

Yair Wiseman
Computer Science Department
Bar-Ilan University
Ramat-Gan 52900, Israel
Tel: 972-3-5317015, Fax: 972-3-7360498
http://www.cs.biu.ac.il/~wiseman
wiseman@cs.huji.ac.il

Keywords: Operating Systems, Graduate Course, Operating System Kernel, Non-Distributed Operating Systems.

Abstract

The use of Non-Distributed Operating Systems is very common and old. Many researchers feel that this field of research is outmoded, and therefore put their efforts into Distributed Operating Systems. Advanced Operating Systems courses generally include an overview of the topical issues of research in the Operating System field. Many instructors prefer using Distributed Operating Systems subjects in order to give their students the contemporary research atmosphere. This encourages graduate students to research Distributed Operating Systems topics. We suggest that Non-Distributed Operating Systems is still an important field worthy of being expanded in graduate courses. An example for such a course is given. This course has been successfully taught in Bar-Ilan University during 2004.

1. Introduction

Advanced operating systems courses are common in many Computer Science departments all over the world. Naturally, academic freedom does not dictate the material taught in such courses. Most courses contain various Distributed Operating Systems materials; e.g. [1,2,3,4]. This induces graduate students to research Distributed Operating Systems. We suggest devising separate courses for Non-Distributed Operating Systems, alongside the existing courses in Distributed Operating Systems. One-processor machines are still the majority of the computing power far and wide. Therefore, a Non-Distributed Operating Systems course can be beneficial for encouraging more graduate students to research this field and to contribute their aptitude. Indeed, some former students of this course have begun researching the subjects described below.

The Non-Distributed Operating Systems is still a vital and dynamic field. A probe of recent operating systems conferences focusing on the "pure" Operating Systems subjects (i.e. Kernel's task) has produced 3 main categories of study in Non- Distributed Operating Systems:

- New and enhanced techniques
- Algorithm improvements
- Statistical studies

We introduce subjects in each category and elaborate on the course material within each subject. The technical depth of this paper will be superficial, because our object does not include specifying technicalities. Instructors who wish to probe into the suggested subjects may look into the cited references. The suggested course integrating these 3 categories has been taught in Bar-Ilan University during 2004. The students were required to do preparatory reading which was followed by a discussion in class. The course was supported by a web site [5].

2. New and Enhanced Techniques

This category contains the enhancements for old and familiar techniques that any Operating System has. The main concepts of the technique are usually taught in the undergraduate Operating System course. In the advanced Operating System course suggested, the up-to-date enhanced techniques are introduced.

2.1. Micro-Kernel

Micro-Kernel [6] is a controversial enhancement for the old concept of the operating system kernel. The first Micro-Kernels were first presented at the beginning of the 70's, but there many implications and ramifications are still being researched nowadays.

The conception of Micro-Kernel is to implement outside the kernel as many functions as possible. This induces a discussion of the advantages and the disadvantages of such an implementation [7,8], of which the main issues are:

- Fault isolation and independence. Kernel malfunction can cause a reboot. With Micro-Kernels less code is executed in Kernel Mode.
- Flexibility. Different strategies and APIs (Application Programming Interface) can coexist when using Micro-Kernels.
- Kernel recompilation is less needed when changes are done in the operating system, because more sections of the operating system are not implemented within the kernel.
- Performance. When using Micro-Kernels, there will be more context switches to the new

operating systems processes (The processes that perform the functions of the Monolithic Kernel).

Another important discussion is which sections of the kernel can be taken out and which sections are essential within the Micro-Kernel. The classic examples of sections that can be taken out are the paging mechanism and the device drivers. Showing the students how this was previously implemented [9,10,11] may offer material for further research.

The opposite approach is to implement inside the kernel whatever possible. This raises the question what definitely should be outside the kernel, or maybe there is no such a component outside the kernel as was suggested in [12]. Such an approach makes the kernel transactions very long. This feature brings up more questions to be discussed in class; e.g. can the kernel be interrupted and if so, when will it be interrupted and by which interrupts [13].

2.2. Super-Pages

Super-Pages is an enhancement of the well-known paging concept. Super-Pages are larger pages that are referenced by the TLB. The internal memory of modern computers has been significantly increased during the last decade. However, the TLB coverage (i.e. the size of the memory that can be referenced directly by the TLB) has been increased by a much lower factor during the same period [14,15]. Therefore, several new architectures like Itanium, MIPS R4x00, Alpha, SPARC and HP PA RISC support multiple page size of the frames referenced by the TLB. In that way the memory size referenced directly by the TLB is higher and the overhead of the page table access time is reduced. In addition, many modern operating systems support Super-Paging.

The Super-Paging concept brings up several questions to discuss in class. First, when should the Operating System upgrade some base pages into a large Super-Page? This dilemma is even more complicated when the processor supports several sizes of Super-Pages; e.g. the Itanium has 10 sizes of Super-Pages. Second, where should the location of the small pages in the memory be? One possibility is placing them in a location that spares the need for relocation of the base page, once the Operating System upgrades base pages into a Super-Page [16]. Another policy is placing the base page in the first vacant location in the memory and relocating it when the Operating System upgrades [17]. Thirdly, who handles the relocation, the hardware or the software [18]?

Some processors and Operating Systems have addressed these questions [19,20,21]. The course shows the students what decisions the specific processors and Operating Systems have taken and what their considerations were. The students are encouraged to express their view and come up with suggestions for improved performance.

2.3. Desktop Scheduling

The schedulers of most of the contemporary operating systems are based on the old well-known schedulers that have been used during the years by the traditional Unixes and other popular operating systems. These schedulers do not always perform well with the new and different characteristic of processes that are used by the new desktop machines.

One of the most notable changes is the multi-media processes that rarely appeared on the old machines and very common nowadays [22]; e.g. Etsion et al. [23,24] explore the effectiveness of the Linux scheduler when using multi-media processes like movie players or games. They show that the Linux scheduler is not tuned well for such processes and they suggest a technique to improve the scheduler in order to get better performance.

Usually, in the undergraduate operating systems course the students are taught about the common schedulers used by the popular operating systems. In the operating systems graduate course they should be able to criticize the timing of those schedulers [25]. Furthermore, they are expected to suggest methods to improve the effectiveness of the original schedulers.

2.4. Versioning File Systems

In 1995, for $200 you could purchase a 0.54GB disk, whereas Slackware Linux 2.2 (Basic Applications+X window) had 0.15Gbytes that make up 28% of the disk capacity. In 2004, for $200 you can purchase a 300GB disk, whereas RedHat Linux Advanced Workstation 2.1 (Basic Applications+X window) for the Itanium Processor has 4.2GB that make up 1.4% of the disk capacity. These facts support the conclusion that nowadays the space pressure on the disk is not high and a portion of the disk can be reserved for backup.

Versioning File Systems are file systems that do not remove the files that have been deleted. The disk retains the blocks of the deleted files and they can be restored if needed. The concept had been previously used by some versions of VMS [26]. However, some new File Systems have been proposed recently e.g. Elephant [27,28] and Moraine [29], which have different policies for different types of files and a better interface for the user.

The students are called upon to explore new deleting timing of the blocks in such a file system [30]. There are some suggestions in the current file systems for this timing. Techniques yielding optimal results should be discussed. In addition, should the policy offered by the file systems be selected by the user or should be automatically selected by the operating system? There is a dispute among the researchers in this field what is better [31] and the students are called upon to take a stand.

3. Algorithms Improvements

Many known algorithms are utilized by the operating system. In this section we suggest showing the students

how these algorithms are adapted by the operating system encouraging improvements on these algorithms.

3.1. ARC

In 1946 Von-Neumann suggested a hierarchy of memories. This concept has been accepted by almost all of the computer manufacturers. In such a hierarchy each of the memories has a greater capacity than the preceding but it is less quickly accessible. When there is no more room in the faster memory, the selecting of the "victim" to be taken out of the faster memory has been traditionally done for decades by the LRU algorithm. The LRU is fast and easy for implementation and has been utilized by many operating systems, but is there a better algorithm?

We suggest introducing new techniques previously suggested such as LRU-K [32], 2Q [33], LRFU [34]. The students should be able to criticize the techniques in several parameters:

- The complexity of the algorithm and time overhead.
- How fast the algorithm can identify “stale” block
- How fast the algorithm recognizes that a block is frequently used and should be in the faster memory.
- The space overhead needed by the algorithm.

ARC is a recently invented method [35,36,37] considered the best known yet. The students may be encouraged to attempt to challenge the method.

4. Statistical studies

Some of the research in the field of operating systems applies to the statistical properties of the operating systems, which can detect common flaws of operating systems and help solve them. In turn we may infer which sections of the operating system are more essential.

4.1. Operating System Bugs

Operating Systems like many other softwares are not bug-free. Some studies have been conducted over the years to analyze bugs on common operating systems like Linux [38,39] and WindowsNT [40]. These studies can help us enumerate the number of bugs an ordinary operating system has. Windows generally has more bugs than Linux. The students may come up with reasons for this.

Another important question is where most of the bugs emerge. The studies show that the device drivers are usually the buggiest section in the kernel. Many developers, who are more familiar with the devices than the kernel, will usually write the Device Drivers. In addition only a few users may have a given device; hence it will be less “battle-tested” than the other sections of the kernel.

The students are shown how this information can help them and how they can improve their products [41]. In complex software systems such information is essential [42]. They may be shown which sections are less trustworthy. In addition, they may be shown common reasons for bugs; e.g. Cut-and-Paste in code writing can be harmful sometimes [43]. Bug lifetime is also an interesting parameter that may show students the standards in the operating system market.

4.2. Benchmarking

Typically, when a product needs to show its performance, it will demonstrate the results by benchmarking. The same usage of benchmarks is done when writing a scientific paper. The benchmarking is selected by the author. This gives the author a wide spectrum of subjective representation of the paper results [44,45].

The Operating Systems field is not different [46]. There are several common benchmarking standards like SPEC [47], but researchers do not tend to take the standards seriously. Many papers contain just partial results and sometimes even results of just one program out of the benchmark suite [48,49]. Moreover, usually authors of papers with the partial results neglect to explain the omission of the missing measurements.

This subject is very important for students who are going to be the next generation of the research community. They should study how to test their ideas with integrity. Hiding the drawbacks and the limitations sets a bad example for students. Honest benchmarking is an essential part of a good paper.

5. Conclusions

The undergraduate course of Operating Systems usually focuses on Non-Distributed Operating Systems, while the graduate Operating Systems course focuses in many universities on Distributed Operating Systems. Many instructors feel that the advanced subjects can be found in the Distributed Operating Systems field, and that the Non-Distributed Operating Systems has very little to offer. This paper has shown that there are enough topics to teach in the Non-Distributed Operating Systems course; thus the conclusion of this paper is that there should be two advanced courses - one on Non-Distributed Operating Systems and one on Distributed Operating Systems. Ignoring the Non-Distributed Operating Systems is actually ignoring most of the computers in the world which are not distributed.

6. Acknowledgments

The author would like to thank Fabin E. Bustamante of Georgia Institute of Technology, Dharmendra Modha of IBM Almaden Research Center, Juan Navarro of Rice University, Yoav Etsion of the Hebrew University, Nicholas Rizzolo of University of Illinois, Daniel Citron of IBM Research Lab in Haifa and Tetsuo Yamamoto of Osaka University.

7. References

[1] K. Schwan, "CS 6210 Advanced Operating Systems", http://www.cc.gatech.edu/classes/AY2004/cs6210_spring/, 2004.

[2] D. Engler, "CS240: Advanced Topics in Operating Systems", http://www.stanford.edu/class/cs240/, 2004.

[3] V. Pai, "COS 518, Advanced Operating Systems", http://www.cs.princeton.edu/courses/archive/fall04/cos518/, 2004.

[4] B. Miller, "CS 736 - Advanced Operating Systems", http://www.cs.wisc.edu/~bart/cs736.html , 2004.

[5] Y. Wiseman, "The Home Page of Advanced Operating Systems Course", http://www.cs.biu.ac.il/~wiseman/2os, 2004.

[6] T. P. Scheuermann, "Evolution in Microkernel Design", Computer Science Department, University of North Carolina, Chapel Hill, NC, 2002.

[7] J. Liedtke, "Toward Real MicroKernels", Communications of the ACM, Vol. 39(9), September 1996.

[8] J. Liedtke, "On Micro-Kernel Construction", Proceedings of the 15th ACM Symposium on Operating System Principles, ACM, December 1995.

[9] D. Golub, R. Dean, A. Forin and Richard Rashid. "Unix as an Application Program", Proceedings of the USENIX Summer Conference, June 1990.

[10] B. N. Bershad, C. Chambers, S. Eggers, C. Maeda, D. McNamee, P. Pardyak, S. Savage and E. Gun Sirer, "SPIN - An Extensible Microkernel for Application-specific Operating System Services", ACM Operating Systems Review, Vol. 29(1), January 1995.

[11] The L4 MicroKernel, http://www.cse.unsw.edu.au/~disy/L4/

[12] T. Maeda, "Safe Execution of User Programs in Kernel Mode Using Typed Assembly Language", Master Thesis, The Graduate School of The University of Tokyo, February, 2002.

[13] G. Anzinger and N. Gamble, "Design of a Fully Preemptable Linux Kernel", MontaVista Software, September 2000.

[14] J. Navarro. Transparent operating system support for superpages, Ph.D. Thesis, Department of Computer Science, Rice University, April 2004.

[15] J. Navarro, S. Iyer, P. Druschel and A. Cox. Practical, Transparent Operating System Support for Superpages, Fifth Symposium on Operating Systems Design and Implementation (OSDI '02), Boston, MA, December 9-11, 2002.

[16] M. Talluri and M. D. Hill, Surpassing the TLB Performance of Superpages with Less Operating System Support, Sixth International Symposium on Architectural Support for Programming Languages and Operating Systems (ASPLOS), San Jose, California, pp. 171-182, October 4-7, 1994.

[17] T. H. Romer, W. H. Ohllrich, A. R. Karlin, and B. N. Bershad, Reducing TLB and memory overhead using online superpage promotion, In Proceedings of the 22nd International Symposium on Computer Architecture (ISCA), pp. 87-176, Santa Margherita Ligure, Italy, June 1995.

[18] Z. Fang, L. Zhang, J. Carter, S. McKee, and W. Hsieh. Re-evaluating Online Superpage Promotion with Hardware Support. In Proceedings of the Seventh International Symposium on High Performance Computer Architecture, pp. 63-72, January 2001.

[19] I. Subramanian, C. Mather, K. Peterson, and B. Raghunath. Implementation of multiple pagesize support in HP-UX, In Proceedings of the USENIX, New Orleans, Louisiana, June 15-19, 1998.

[20] N. Ganapathy and C. Schimmel, General Purpose Operating System Support for Multiple Page Sizes, In Proceedings of the USENIX, New Orleans, Louisiana, June 15-19, 1998.

[21] S. Winwood, Y. Shuf, H. Franke, Multiple Page Size Support in the Linux Kernel, Ottawa Linux Symposium, Ottawa, Ont, Canada, June 2002.

[22] J. Nieh, J. G. Hanko, J. D. Northcutt, and G. A. Wall, "SVR4 UNIX Scheduler Unacceptable for Multimedia Applications", Proceedings of the Fourth International Workshop on Network and Operating Systems Support for Digital Audio and Video, Lancaster, UK, pp. 35-48, November 1993.

[23] Y. Etsion, D. Tsafrir, and D. G. Feitelson, "Desktop Scheduling: How Can We Know What the User Wants?". In the 14th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), June 2004.

[24] Y. Etsion, D. Tsafrir, and D. G. Feitelson, "Human-Centered Scheduling of Interactive and Multimedia Applications on a Loaded Desktop". Technical Report 2003-3, School of Computer Science and Engineering, The Hebrew University of Jerusalem, March 2003.

[25] A. Goel, L. Abeni, C. Krasic, J. Snow, and J. Walpole, "Supporting Time-Sensitive Applications on a Commodity OS", Fifth Symposium on Operating Systems Design and Implementation (OSDI '02), Boston, MA, December 9-11, 2002.

[26] M. D. Schroeder, D. K. Gifford and R. M. Needham, "A Caching File System for a Programmer's Workstation", Proceedings of the tenth ACM symposium on Operating systems principles, Orcas Island, Washington, pp. 25-34, 1985.

[27] D. S. Santry, M. J. Feeley, N. C. Hutchinson, A. C. Veitch, R. W. Carton and J. Ofir, "Deciding When to Forget in the Elephant File System", ACM Symposium on Operating System Principles, Kiawah Island Resort, South Carolina, pp. 110-123, December 12-15, 1999.

[28] D. J. Santry, M. J. Feeley, N. C Hutchinson, and A. C. Veitch, "Elephant: The File System That Never Forgets", Proc. Workshop on Hot Topics in Operating Systems, Rio Rico, Arizona, pages 2-7, 1999.
[29] T. Yamamoto, M. Matsushita and K. Inoue, "Accumulative Versioning File System Moraine and Its Application to Metrics Environment MAME", Proceedings of the 8th ACM SIGSOFT international symposium on Foundations of software engineering, pp. 80-87, Shelter Island, San Diego, California, USA, November 6-10, 2000.
[30] C. A. N. Soules, G. R. Goodson, J. D. Strunk, and G. R. Ganger, "Metadata Efficiency in Versioning File Systems," in Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST 2003), pp. 43-58, San Francisco, California, March 31-April 2, 2003.
[31] K. Muniswamy-Reddy, C. P. Wright, A. Himmer and E. Zadok, "A Versatile and User-Oriented Versioning File System", in Proceedings of the Third USENIX Conference on File and Storage Technologies (FAST 2004), pp. 115-128, San Francisco, California, March 31-April 2, 2004.
[32] E. O'Neil, P. O'Neil and G. Weikum, "The LRU-K Page Replacement Algorithm for Database Disk Buffering", Proceedings of SIGMOD `93, Washington, DC, May 1993.
[33] T. Johnson and D. Shasha, "2Q: a low overhead high performance buffer management replacement algorithm", Proceedings of the Twentieth International Conference on Very Large Databases, VLDB' 94, Santiago, Chile, pp. 439-450, September 1994.
[34] D. Lee, J. Choi, J.-H. Kim, S. H. Noh, S. L. Min, Y. Cho, and C. S. Kim, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," IEEE Trans. Computers, vol. 50, no. 12, pp. 1352-1360, 2001.
[35] N. Megiddo and D. S. Modha, "ARC: A Self-Tuning, Low Overhead Replacement Cache," Proc. of the 2nd USENIX Conference on File and Storage Technologies (FAST'2003), San Francisco, pp. 115-130, March 31 - April 2, 2003.
[36] N. Megiddo and D. S. Modha, "One Up on LRU" ;login: - The Magazine of the USENIX Association, vol. 28, no. 4, pp. 7-11, August 2003.
[37] N. Megiddo and D. S. Modha, "Outperforming LRU with an Adaptive Replacement Cache Algorithm," IEEE Computer, pp. 4-11, April 2004.
[38] A. Chou, J.-F. Yang, B. Chelf, S. Hallem, and D. Engler. "An Empirical Study of Operating Systems Errors", In Proceedings of the 18th ACM Symposium on OS Principles (SOSP), pp. 73-88, Lake Louise, Alta, Canada, October 2001.
[39] D. G. Majors, "An Investigation of the Call Integrity of the Linux System", 14th. IEEE International Symposium on Software, Reliability Engineering ISSRE 2003, Denver, Colorado November 17-20, 2003.
[40] J. Xu, Z. Kalbarczyk, and R. Iyer, "Networked Windows NT System Field Failure Data Analysis," Proc. 1999 Pacific Rim Int'l Symp. Dependable Computing, IEEE CS Press, Los Alamitos, CA, 1999.
[41] M. M. Swift, B. N. Bershad, and H. M. Levy, "Improving the Reliability of Commodity Operating Systems", Proceedings of the 19th ACM Symposium on Operating Systems Principles, Bolton Landing, New York, October 19-22, 2003.
[42] N. E. Fenton and N. Ohlsson, "Quantitative Analysis of Faults and Failures in a Complex Software System," IEEE Transactions on Software Engineering, 26(8), pp. 797-814, August, 2000.
[43] Z. Li, S. Lu, S. Myagmar, and Y. Zhou, “CP-Miner: A Tool for Finding Copy-paste and Related Bugs in Operating System Code”, Sixth Symposium on Operating Systems Design and Implementation (OSDI '04), San Francisco, December 6-8, 2004.
[44] X. Zhang, "Application-Specific Benchmarking", Ph.D. Thesis, Harvard University, May 2001.
[45] M. Seltzer, D. Krinsky, K. Smith, X. Zhang, "The Case for Application-Specific Benchmarking", In Proc. HotOS-VII, pp. 90-95. Rio Rico, AZ, March, 1999.
[46] J. C. Mogul. , "Brittle Metrics in Operating Systems Research", In Proc. HotOS-VII, pp. 90-95. Rio Rico, AZ, March, 1999.
[47] Standard Performance Evaluation Corporation (SPEC), http://www.spec.org
[48] D. Citron, "MisSPECulation: partial and misleading use of spec CPU2000 in computer architecture conferences", Proceedings of the 30th Annual International Symposium on Computer Architecture, pp. 52-59, June 9-11, 2003.
[49] D. Citron, J. Hennessy, D. Patterson, G. Sohi, "The Use and Abuse of SPEC", Proceedings of the 30th Annual International Symposium on Computer Architecture, pp. 73-77, June 9-11, 2003.