Windows 2000 Research Edition

Where the Academic Knights meet the Evil Empire...

Werner Vogels

The rivalry in the operating system market place has a severe impact on the academic world. Where in the old days intellection quality and careful deliberation would prevail, nowadays discussions about operating systems research appear to be more like the battlefield of a holy war, with objectivity as its main victim. We have tried to side step the emotional current, and select an operating system that could bring our research into the next century, based on objective technical and organizational criteria. This paper describes how this evaluation lead to the insight that Microsoft's Windows NT is the operating system that is best prepared for the future.

Introduction

Until recently there was no doubt in academia which operating system to use for systems research: Unix, whether it was a BSD or System V derivative, was the predominant choice. Unix, which had its roots in research, was used since its inception to investigate fundamental system research, and the accumulated knowledge in academia about its internals and operations was significant. Other available operating systems such as VMS and MVS, had their roots in the commercial world and knowledge about these systems never accumulated to the critical mass were these systems could be considered for widespread research tasks. Although new research operating systems have been developed, none have found the following that the established Unix's received. Solaris, Linux, FreeBSD and others continue to dominate the academic landscape, but slowly but surely Windows NT is now entering the academic world as a viable, alternative platform for research.

Although academia looked with fascination at Dave Cutler's attempt to build a new operating system from the ground up [Custer 95]. All expected that Windows NT would go the same way as the other commercially designed operating systems before it and remain in the dark corner from a research use point of view.

About four years ago, not long after the final major release of academic version of the Unix operating system (BSD 4.4) [McKusick et al. 96], the farewell of the Berkeley systems group and the early demise of Mach as the last of the research operating systems, the operating system research world was at a crossroads. Intel based personal computers were becoming ubiquitous, and a myriad of Unix operating systems was available for this platform. Eventually many moved to use Linux, a popular architectural clone of the traditional Unix. At the Computer Science department at Cornell University we made the decision to conduct our research on Windows NT. By that time we had learned enough from the early design of Windows NT to realize that it was a major step forward in operating system design. It would provide us with a platform on which we could perform research more effectively and it would allows us to focus on the future directions without having to worry whether the operating system was capable of supporting innovation.

By now our complete educational operation and the majority of our research projects have switched to using Windows NT, or Windows 2000 as it now officially has been christened [Solomon 98]. The ride has been rocky and fascinating, but certainly rewarding. In this article I want to share some of the reasoning behind our choice for Windows NT and to share some our experiences with Windows NT as a research platform.

OS research as religion

The biggest hurdle in starting research on Windows NT was not technical. It was to overcome the skepticism of our colleagues who were convinced that it would not be possible to use Windows NT as a good platform for research. The predictions were fascinating: we would turn into a bug-fixing factory, Microsoft would sue the department for every technical publication, Microsoft would hide the pieces of buggy code from us or Bill Gates would personally tell us where and how we should do our research [Welsh 99].

The operating systems research community has not remained untouched by the market place rivalry between Microsoft and the group lead by Sun Microsystems. It is even more unfortunate that the positions taken are not based on intellectual deliberation but purely on emotional grounds. Many see Microsoft

Werner Vogels is a research scientist at the Department of Computer Science of Cornell University. His research targets high-availability in distributed systems, with a particular focus on enterprise cluster systems. He is co-founder and vice-president of Reliable Network Solutions, Inc., which specializes in building solutions for very large-scale reliable distributed systems. He is co-chair of the 1999 Usenix Windows NT Symposium. His personal homepage is at http://www.cs.cornell.edu/vogels

as the Evil Empire, out to squash every attempt at innovation, and working with them is seen as collaboration with the enemy of free academic speech. The pros and cons are often discussed with a righteous zeal that is frightening.

Our own experiences with Microsoft can only be described as extremely positive, never before have we had such a positive relation with a vendor, without any pressure from their side. We can only conclude that the reasons for the controversy must be found in a sort of traditional emotional bonding of academia with the "underdog" and that no real experiences drive the discussion.

Gaining knowledge

The foremost reasons why Unix was such a powerhouse in operating system research was the great amount of knowledge accumulated over the years about the internal operation of the operating system. Many of us had become gurus about some part of the OS kernel and could recite the fields of an I-node structure at late night meetings or discuss which data structures to modify to add a new protocol at runtime over an early morning cappuccino. Many of us were and still are afraid to leave this bastion of safety behind and trade it in for working on an operating system that at first sight had nothing in common with our beloved Unix. And our annotated version of the Unix version 6 code [Lions 96] wouldn't be of much help any more either…

It took more then a year of immersion in the technology to get a level where I felt confident again to direct others in our research group. Together with the overall organizational issues I think we lost one and a half year worth of research time to make the switch in the most fundamental way. Others are making the switch more gradually and are experiencing a more smooth transition.

All Operation Systems are created equal

Our experiences with switching to Windows NT have made us somewhat more philosophical about the nature of operation systems. The most fundamental observation is that, when stripped to their core, all operating systems are equal. The functionality of the Windows NT kernel is just as all other kernels: it abstracts the hardware in the usual sense: processor, process and threads hide the CPU complexity, volumes, file systems and files hide the storage devices, protocols hide the network, shared memory and messages are used to allow sharing of resources.

What we often call operating systems has nothing to do with the real core of the system. Unix for most of us is a collection of shell commands and development libraries. David Korn's UWIN [Korn 97] and Softway's Interix [Walli 97] both show that you can give users and developers a 100% Unix experience including X-windows and multi-user rlogin servers, while running on an Windows NT kernel.

Windows NT for most of us is the Windows Explorer and point-and-click, and according to Microsoft it includes a web browser. Although I have not seen a complete re-implementation of the Explorer for Unix, the Win32 compatible libraries from Mainsoft [MainWin] used in the port of Internet Explorer show that you do not need a Windows NT kernel to get to the same user experience.

Many see the rich Win32 programming interface as the native programming model for Windows NT, and although most Windows applications are designed using this interface, it is not the Windows NT kernel interface. Almost no applications are built using the kernel interface, and you would have a hard time finding the complete documentation for all the system calls. Describing Windows NT as a micro-kernel, feels awkward, as the kernel is certainly not small, but it is does describe the abstraction correctly in which the kernel provides base services and the specific application context is provided through subsystem servers or personalities. Win32 is one of the personalities running on top of Windows NT, OS/2 and Posix are others delivered by Microsoft. One can run Windows NT without these standard personalities and build your own.

What is an Operating System?

This question seems to be on the mind of many people these days, infused by the Microsoft Trial. Academics in general have taken a very narrow view of what an operating system is. David Faber at Microsoft trial defined an operating system as "*the software that controls the execution of programs on computer systems and may provide low-level services such as resource allocation, scheduling and input-output control in a form which is sufficiently simple and general so that these services are broadly useful to software developers*" [Faber 98].

In research community this strict distinction serves to distinguish the "real men" from the "boys". Researchers and hackers that work in the area defined by this narrow definition of operating systems, consider themselves part of the select circle of people working on the core of the systems area of computer science. Once you are in this circle you will become part of the secret society that practices the black art of OS research and will start to regard any other activity of systems development as irrelevant to the future of computer science.

For a long time the line was drawn at the kernel / user space boundary, and one could only consider himself a true OS researcher after having developed at least two device drivers and hacked on the terminal driver of the BSD 2.9 kernel code. In modern operating systems such as Windows NT, the notion of where exactly operating systems services are located is not that simple any more. Fundamental services are split between kernel and user space in attempts to optimise their efficiency and avoid uncontrolled growth of kernel services.

The pervasiveness of distributed services in modern systems can be considered a threat to the traditional notion of operating systems. Many support services are required to make distributed systems work efficiently and effectively and these services, such as security and directory services or distributed object support and cluster management, are not part of a traditional view of operating systems, but they are essential to the operation of modern operating systems. This results in that an operating system no longer is a simple division between kernel and user space, but consist of a myriad of services, of which some are kernilized, some are local and others are remote. Operating systems that address the needs of current and future clients and servers no longer span a single computer and they abstract services away from physical nodes allowing user to be part of a larger, potential global operating environment.

Will the real Dinosaur please come forward?

Until the spring of 1995 we were deeply committed to SunOS (and other BSD derivatives). At that moment its vendor was discontinuing the operating system, and had designated Solaris, which had its root in AT&T's System V as the successor. This event forced us to take a step back and evaluate our research directions and our expectations with respect to the operating systems to use.

If one issue in our discussions was dominant, it was the fact that most of the operating systems we were looking at were actually very old fashioned, in design, in structure and in implementation. Most of these operating systems had their conception in the 1970s and did not change much in structure since then. Linux could be seen as an exception since it was developed in the second half of the 1980s, but its structure mirrored that of the traditional Unix systems, and as such it could be considered one of them.

The significant advances made in academic computer science, in OS research and in system software engineering, have had only minimal impact on the design and implementation of commercial operating systems. The design of all Unix systems violates almost all of the software engineering principles presented to first year's Computer Science students. The design is monolithic with almost no modular structure, and the internal kernel interfaces are not strictly enforced which introduces dependencies on the actual implementation of data structures, making it impossible to upgrade or replace modules without also redesigning several other modules. For example to replace the scheduler in any of the BSD's one needs to spend two weeks searching for all dependencies and fixing other sources.

At the top of our long wish list for an ideal research operating system, were three important general points:

1. The design and implementation of the operating system should comply with modern software engineering principles, allowing researchers to introduce new components, and replace core components without redesigning the complete system.
2. The overall structure of the operating system, user and kernel space components, should be designed towards the future: distribution and multi-computer awareness, for example, should be pervasive throughout the whole system.
3. The operating system vendor should be open to innovation. Our experiences in the past had been that vendors always ignored important research results and only followed very narrow paths of incremental improvements.

Windows NT was the only operating system that came close to matching most of our requirements, with a handful of operating systems such as QNX and Utah's OS-kit trailing close. None of the Unix based operating systems came close to fulfilling our requirements. As noted before the core of those operating systems is based on 20–30 year old designs and these operating systems still treat computers as single entities without a coherent, integrated distributed approach.

Although from a 30,000 feet high Windows NT looked like the proverbial dinosaur, a closer look revealed a truly modern operating system. Object oriented design is pervasive through the system including the kernel, there is a complete distributed strategy with at its core a distributed object technology and includes a complete integration of distributed services such as security, directory, message queuing, distributed transactions etc. And last no but least, there is a real desire by the vendor to continuously innovate its operating system and the overall services. Microsoft doesn't hesitate to incorporate academic results into operating system, and is open for new directions.

Innovation as a life style

Microsoft is not conservative in its OS development. While most vendors only consider changes to their core OS services under extreme market pressure, the core of Windows NT has changed significantly over the past years to accommodate the demands of modern computing. Especially the upcoming release of Windows 2000, formerly known as Windows NT 5, makes that the Microsoft takes the operating system functionality to "the next level".

The advances in Windows 2000 are too numerous to enumerate here: They range from a file system cache for disconnected operation, which was originally developed at CMU in the CODA project; to a remote storage service that automatically moves old data from your hard disk to remote servers if you are running out of disk space, from tight security integration, with Kerberos (developed at MIT) as the dominant security provider, to a complete integration of network quality of services tools including data transmission shapers and priority scheduling and queuing, and from attributed based distributed component programming to indexing support integrated in the file system [Microsoft].

We are witnesses of a unique process: never before have we seen such a radical overhaul of an operating system targeted for the enterprise market. In general this market is very conservative and not interested in taking risks. However the problems of scale, management and distribution are asking for radical solutions to get to a computing base that can bring us into the next century. One of the markets where we will see the main competitive battle between Microsoft and others will be that of the E-Commerce servers. Web farms with hundreds of nodes, with support services for load balancing, replication, light-weight transactions, in-memory database caches, distributed and single image management, etc., are really pushing the envelope of all operating systems that are currently on the market. Windows NT is still considered to be the new kid on the block in the Internet services world, but it is clear that the risks that are taken now are the right moves to prepare the operating system for operation in these emerging massive computing environments.

The Bugs

Innovation comes at a price. One of the costs of introducing a significant amount of new code is the number of software defects per lines of codes increases. While measurements actually let us believe that Microsoft products are quite reliable at

.5 defects per KLOC (thousand lines of code), introduction of new, fresh code has a disastrous effect on this number. The outlook becomes even more worrisome when we realize that Microsoft is not only introducing new code, but is also changing all of its old code. An automated process is converting all of the Windows NT code to be 64 bit safe. This process converts 30 thousand lines of code per day and is believed to catch all pointer arithmetic cases.

An important question is whether the introduced functionality is worth the unavoidable initial instability that is bound to occur. Whenever taking risks to achieve major improvements, there is always the down side that there is some change of failure and it is likely that we will see a number of components of NT coming under intense scrutiny from industry and academia. Some components, such as the directory services, may become a performance bottleneck in the overall distributed operation, or the wide spread security integration could introduce a critical dependency on the high-availability of the security servers.

From a research point of view, these problems do not really bother us. The advantage of performing research on a system, which has distribution at its core greatly outweighs the consequences of working with a cutting edge operating system. However I must admit that at more then one occasion my students had to control their murderous intentions towards the IIS or MTS developers or were they kept their good spirits by contemplating the horrible tortures one could perform on the person that had designed the COM security architecture.

Windows Research

There are some properties of Windows NT that make it particularly suitable for research purposes. The operating system kernel for example is designed with extensibility in mind, to allow developers of hardware based services, new protocols and file systems to add their functionality to the system without much effort. All kernel code is developed following a strict object oriented paradigm and its functionality can only be accessed through interfaces, none of its implementation is visible.

One of the designs abstractions of the Windows NT kernel I find it particularly fascinating to work with is the device object. A device object in an instance created by driver objects, which encapsulates a unit of kernel based software, whether this is a device driver, a network protocol or a file system filter. These objects have the interesting property that they can be “attached” to other device objects, and as such can intercept and manipulate all requests flowing to and from the original device object. This way it is relatively simple to add for example a file system object that compresses or encrypts data before the data reaches the under laying file system, to redirect disk requests to a replication volume, or to trace device object interaction during development phases.

The strict object oriented approach is very well done from a design point of view, but every old-style hacker’s heart starts bleeding when he or she realizes that he can no longer do a quick fix, inspect a few data structures and secretly swivel some pointers to make things work better or make more informed decisions. The internal kernel interfaces are elaborate, but it appears there are always some things one cannot do as efficient as possible. However, in four years of NT kernel hacking only on one occasion we needed to break through the standard kernel interface: we wanted to add a fast trap into the kernel for fast user-level protocol processing, and the pages which hold the trap dispatch tables were protected after the system boot.

Another example of what makes Windows NT particular suitable for research is the fundamental manner in which advanced distributed services are integrated into Windows NT. It allows us to rely on ubiquitous support services and concentrate on advancing the state of the art where it is really needed. Windows NT Security provides a complete set of services integrated into all sections of the operating system. Researchers who are developing an advanced multi-node replicated transaction server can use off-the-shelf authentication, authorization, key management, data signing, and encryption mechanisms into their system without much pain.

The use of the COM object model in all the Windows NT services allows research projects to import these services in a very simple manner. The existence of COM makes it trivial for research projects to export their interfaces in a language independent manner. The Ensemble project for example has developed a protocol environment for distributed operations in the ML programming language, and by using a COM interface are the services offered by Ensemble available to C++, Java and VB programmers. This allowed the researchers to side-step the time consuming development of native language interfaces.

It helps of course to have all the tools, applications, servers, operating system versions and their source code available. Microsoft is very generous to academia and makes all their tools from operating systems to compilers, including tons of documentation as well as subscriptions to the developer network, available to the departments free of charge.

Source code availability turned out to be not crucial, and was only once used to make actual changes to the operating systems [von Eicken et al. 97]. The source is extremely useful as additional documentation, to examine unexpected behaviour or to provide templates for similar projects. As one can perform complete source code level debugging of all parts of the operating system including the kernel, source codes helps us to develop experimental services faster and in tune with existing functionality. Students are free to work with the source code and are not prohibited in any way from applying the knowledge they gained in their later careers.

Interactions with the Evil Empire

Microsoft realizes the potential of widespread adoption of Windows NT for research purposes and there is dedicated academic relations team whose single task it is to facilitate the technology transfer between Microsoft and academia and vice versa. Source licensing is very liberal compared to other OS vendors and several institutions are involved in active exchanges with product and research groups within Microsoft. Joint projects are in progress, joint papers are starting to appear and academics frequently present cutting edge result to Microsoft developers and researchers.

There is a direct impact of academia on Microsoft products, through involvement in the strategy phases of products as well as through academic knowledge transfer into products and design groups. Microsoft also provides research funding for some relevant groups and fellowship and research internships for students.

Summary

Four years of research on Windows NT have taught us that we made the right choice in leaving the Unix behind. Windows NT is an exiting, modern operating system, years ahead of its competition, in design, in software engineering, in its implementation and in the actual services offered. It took quite some time to reach the same level of knowledge and insight we used to have of Unix systems, but now that we have arrived at that same knowledge point, is it clear that our research is making progress faster than ever before.

Working with Windows NT requires certain level of resilience, not because of flaws in the operating system, but because of the zealous attacks by colleagues and other researchers. Publishing papers about research performed on Windows NT is still quite difficult as many of our peer still believe that no good research can be performed on Windows NT. We hope that eventually the advanced technical nature of the operating system will prevail in the discussion, and that we can have a community where research results can be shared without sarcasm or the risk of igniting yet another holy war.

References

[Custer 95]
Custer, Helen, "Inside Windows NT", Microsoft Press, Seattle, 1995, ISBN 1-55615-481-X

[Faber 98]
Faber, David, "Expert Testimony of Professor David J. Farber", Department of Justice, Anti Trust Division, December 7, 1998, http://www.usdoj.gov/atr/cases/f2000/2059.htm

[Korn 97]
Korn, David G.,"UWIN – UNIX for Windows", The USENIX Windows NT Workshop 1997, August 11–13, 1997, Seattle, Washington, see also http://www.research.att.com/sw/tools/uwin

[Lions 96]
Lions, John, "Commentary on UNIX 6th Edition with Source Code", Peer-to-Peer communications, 1996, ISBN 1-57398-013-7.

[MainWin]
MainWin, a MainSoft product, http://www.mainsoft.com

[McKusick et al. 96]
McKusick, Marshal Kirk, Keith Bostic, Michael J. Karels, John S. Quarterman, "The Design and Implementation of the 4.4 BSD Operating System", Addison-Wesley, 1996, ISBN 0-201-54979-4

[Microsoft]
Microsoft Corporation, "What's New in Windows 2000 Server" http://www.microsoft.com/ntserver/windowsnt5/exec/overview/WhatsNew.asp

[Solomon 98]
Solomon, David, "Inside Windows NT, second Edition", Microsoft Press, Seattle, 1998, ISBN 1-57231-677-2

[von Eicken 97]
von Eicken, Thorsten, Brian Bershad, Geoff Lowney, Todd Needham, Margo Seltzer, Nick Vasilatos and Werner Vogels, "Do you need source with that", panel at the USENIX Windows NT Workshop 1997, August 11–13, 1997, Seattle, Washington. Summary in Usenix Login, November 1997

[Walli 97]
Walli, Stephen R."OPENNT™: UNIX Application Portability to Windows NT™ via an Alternative Environment Subsystem", The USENIX Windows NT Workshop 1997, August 11–13, 1997, Seattle, Washington, see also http://www.interix.com

[Welsh 99]
Welsh, Matt, "Boycott Microsoft, Protect the Future of Computing Technology", http://www.boycott-ms.org/ last updated 10 Feb. 1999