Specialized Execution Environments

Maria Butrico, Dilma Da Silva, Orran Krieger[†], Michal Ostrowski[†], Bryan Rosenburg, Dan Tsafrir, Eric Van Hensbergen, Robert W. Wisniewski, Jimi Xenidis

IBM Research

Virtualization has become popular (again) as a means of consolidating multiple operating systems (OSes) onto a smaller set of hardware resources. The roles of OSes in such environments have changed. Whereas normally an OS provides balance between the demands of application and hardware support, in the world of virtualization it can be beneficial to split these roles. One OS may support a particular application set and use other OSes to interact with physical hardware. The hypervisor, or virtualization layer, provides communication facilities for the inter-OS communication needed to support such a deployment model.

OSes can now be (1) dedicated to service specific applications, (2) detached from the underlying hardware, and (3) releaved from the need to provide the entire legacy support normally required of a generic OS. A benefit is that the OS can be saved, restored, and migrated. Further, the OS can be customized to needs of that particular application without the need to support legacy interfaces. Conversely OSes bound to physical hardware for the sake of providing virtual I/O services do not need to support application environments. This split enables deploying the right OS for the right task. It allows innovation in the kernel because the OS responsible for providing the application environment is not required to provide legacy support. Instead a side OS can be called upon for such non-performance critical functionality. OSes can be updated and maintained independently; an OS security or bug fix update needed by a particular application need not affect OSes that are dedicated to providing hardware support.

We argue that this separation of responsibilities should be encouraged and that the next step is not just to deploy of the right OS for the right task, but also to develop the right specialized OSes/programs to fulfill these tasks. We envision specialized programs that replace general purpose OSes in these roles. Taken a step further, the overall performance of a system may be improved if these specialized programs can be made to run in execution environments that are less complex and demanding than those needed to run general-purpose multi-tasking OSes. Increased simplicity would also have positive implications on security and reliability issues. We envision the existence of a small hypervisor running a combination of partitions and dedicated programs inside specialized execution environments (that are something less than a full OS partition). No single label applies to all parts of this vision. “Virtualization”, “monolithic kernel”, “micro kernel”, and “exo-kernel” all have their places. Our vision is pragmatic in that it permits a system to be a melting pot of all of these ideas, choosing the right model for each task. Below we present four examples from our own work that provide evidence and motivation for deploying specialized execution environments. While the results from these four efforts are still evolving, the performance and functionality advantages we have observed demonstrate the value of such environments.

[†]Currently at VMware.

High Performance Computing Applications

The original motivation for using specialized execution environments for high-performance applications came out of our work on the DARPA High Productivity Computer Systems program. During our analysis of HPC workloads, we often observed that the applications ran significantly faster when run stand-alone in our simulation environment than when they were executed on top of an operating system in the same simulation environment. We reasoned that if we could “push the operating system out of the way” during execution, we could increase the performance and reliability for certain applications. To test our hypothesis, we ran a sparse-memory benchmark on an extremely thin stand-alone kernel within a partition on hardware. The stand-alone kernel was single threaded and used a flat address space, communicating results over a shared-memory segment to a peer partition running Linux. As reported in [6], the overhead of virtualization results in better run times for Linux at small workload sizes, but as the size of the workload grows, the stand-alone application kernel demonstrates an increasing performance advantage for the same operations. The use of specialized execution environments to control OS-related performance perturbation has dramatically improved performance, as reported in [5].

Commercial Workloads

Many applications have components that do not require the full-fledged support provided by general-purpose operating systems. By offloading the execution of performance-critical components onto specialized execution environments we may find opportunities to accelerate applications and simplify system management. Acceleration is enabled by tailoring operating system services to match application-specific characteristics, exploring performance optimization opportunities that are not present with general-purpose service implementations. Management simplification arises from the ability to exploit additional hardware resources without incurring the costs of managing additional general-purpose operating system images.

Many middleware applications and environments provide duplicate implementations of operating system services. For instance, Java Virtual Machines provide their own implementations of scheduling, networking, and memory management. Databases often include their own storage and network ”drivers” as well as their own authentication facilities and file systems. Specialized execution environments can be used to avoid duplicated functionality in the system software stack, allowing application code to run at a level much closer to the underlying hardware.

We demonstrated the suitability of this software structuring approach in achieving performance and management advantages by implementing a specialized execution environment, called J9/Libra, to run Java-only (non-JNI) applications on a cluster of blades [1]. The J9/Libra execution environment offers the JVM the mechanisms it needs to enforce its own resource management policies, eliminating issues such as JVM safe-point scheduling and double virtualization (address translation done first by hardware and software, and then by JVM read/write barriers). We also specialized the execution environment to meet the requirements of our target application, the Nutch open-source search engine [3]. With our initial J9/Libra prototype the performance of the Nutch query server is 30% better than the same codebase running on a Linux 2.6.17 Xen environment. The recent product announcement from BEA [4] is further evidence of ongoing work on pursuing JVM optimizations by taking the operating system out of the picture. Recently the Libra execution environment has been ported to x86 and extended to support C/C++/glibc environments.

Driver Partitions

When one considers virtualized environments based on the IBM Power Hypervisoror Xen [2], each piece of I/O hardware is assigned to a single OS (unless the hardware itself is self-virtualizing). Such OSes may then be tasked with providing virtual I/O services to other OSes that do not have physical device access using communication services facilitated by the hypervisor.

A device driver in a general-purpose OS (e.g. Linux) must gracefully co-exist with a multitude of competing OS services. Supporting such generality requires that the OS be structured to allow for preemption, multi-threading, and scheduling priorities, all of which represent overhead code that is not actually necessary to run the intended services.

On the other hand, the functionality inherent in device drivers and virtual device services does not depend on a full OS environment. The role of such code is to relay and transform requests and completions between a virtual-device service and a device driver. Simplistically, one could implement such a service using a single, non-preemptible thread that polls (and potentially blocks on) an event notification interface and thus can be implemented in an environment that does not have all of the preemption and MMU management features required of a general purpose OS. (As demonstrated by user-space device drivers.)

Writing the code for such dedicated “OS”es or programs does not necessarily require recreating device drivers. Most device drivers in Linux depend on a relatively small set of interfaces, and our previous work in this space leads us to believe that it is feasible to re-use Linux device drivers in code bases other than the Linux kernel proper, without substantial device driver changes [7]. Once we have such dedicated programs and services, it becomes reasonable for the hypervisor to provide a specialized environment to run them. By limiting the functionality available in a device-support environment, it becomes possible to take advantage of CPU-specific features to lessen memory and/or TLB footprints and lower context switch costs.

Our ongoing investigation in this space (on PowerPC) show that these two techniques combined can eliminate 70% percent of the CPU overhead that I/O virtualization introduces.

Scalability and Hardware Support

Application software stacks have not been designed to achieve the level of scalability to be offered by future multicore architectures. Specialized execution environments can help solve this problem by using a partition with the exact resources required for optimal execution of a specific application. These customizations may not be feasible in a general purpose OS that must simultaneously support the large-scale application in question alongside other generic processes. In essence, the specialized execution environment can be viewed as an accelerator where the cores used to execute the application are not explicitly known or managed by the originating OS.

Conclusion

This paper presents our vision, as well as some of our experiences and ideas related to virtualization-enabled specialized execution environments. While the results are still preliminary, the performance and functionality advantages we have observed have convinced us of the fundamental value of such environments. Specialized execution environments provide an exciting venue for OS research that has long been stifled by the complexity of introducing any innovation into general purpose OS stacks. The specialized execution environment approach also introduces new challenges related to identifying the proper interfaces, communication primitives, and interactions between independently developed and deployed special-purpose OSes.

References

[1] G. Ammons et al. Libra: a library operating system for a jvm in a virtualized execution environment. In *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*, 2007.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the Symposium on Operating System Principles*, Bolton Landing, New York, U.S.A., 2003.

[3] M. Cafarella and D. Cutting. Building Nutch: Open source search. *Queue*, 2(2):54–61, 2004.

[4] G. Clarke. BEA adopts virtual strategy with VMware. *The Register*, December 2006.

[5] E. V. Hensbergen. The effect of virtualization on OS interference. In *Proceedings of the 1st Annual Workshop on Operating System Interference in High Performance Applications*, August 2005.

[6] E. V. Hensbergen. Partitioned reliable operating system environment. *Operating Systems Review*, 40(2), April 2006.

[7] O. Krieger et al. K42: Building a complete operating system. In *Proceedings of EuroSys'2006*, pages 133–145. ACM SIGOPS, April 2006.