

Exokernel: An Operating System Architecture for Application-Level Resource Management

Summary

This paper proposes another approach to operating system design that improves both efficiency of hardware resources utilization and abstraction : the exokernel architecture. The exokernel is a small kernel that provides low-level interfaces to access to hardware resources. It helps applications control over hardware resources more directly by providing a low-level interface to applications. To test the idea, a prototype exokernel operating system were created . Outstanding results were achieved on both efficiency and performance according to measurements.

Key ideas

The exokernel follows these design principles: Securely expose hardware, Expose allocation, Expose names and Expose revocation. It leaves policy decisions for library operating systems to decide.

The key idea of exokernel architectural design is to separate protection from management by exporting hardware resources, not emulating them like in traditional operating systems. 3 techniques were used to securely export hardware resources: *secure binding*, *visible resource revocation* and an *abort protocol*.

Secure binding separate authorization from utilization of resources, improving performance. This technique is implemented using hardware, software caching and downloading code.

Visible resource revocation gives library operating systems and applications more control over resource allocation and deallocation and inform them about the state of hardware resources.

The abort protocol deal with library operating systems that fail to satisfy the revocation protocol, in this case it break all secure bindings with the uncomplying library operating system and inform it.

Background

Traditional operating systems provide access to hardware resources through a set of abstractions such as file system, processes, interprocess communication. The UNIX architecture is an example where those abstraction is exported through a system-call interface to the kernel. Those system-call are hard-coded to the kernel thus very difficult to extend or modify. The approach of forcing applications to use fixed high-level abstractions compromises both of their performance and functionality. Microkernel systems address some of those problems by minimizing the amount of kernel abstraction but still has the downsides of having fixed abstractions. Many similar research has been done to improve extensibility of operating system but none achieved the same degree as the exokernel.

Contributions

This paper proposes a design for operating systems that focus on separating protection and management. This approach greatly improves flexibility in operating systems, allowing applications and libraries to implement kernel-level abstractions like Interprocesses Communication and File system. Performance is also improved as low-level hardware interactions can be implemented efficiently.

The Aegis exokernel and the ExOS library operating system is also presented in this paper. They are used to experiment with the exokernel system design.

Critique

Although theoretically, performance could significantly increase following the exokernel approach, implementing functioning library operating systems and applications based on those principles is extremely difficult as they have to conform to many standards and require deep understanding of hardware functionalities.

This paper does not state clearly what type of system could use the exokernel structure, for example general computing systems or more specialized ones.