Shashi Shivaraju
XID:C88650674

Paper Review 1 - Exokernel: An Operating System Architecture for Application-Level Resource Management

This paper describes an operating system architecture called Exokernel, which provides control over machine resources to an un-trusted library operating system by defining a low-level interface in order to achieve application level resource management. Library operating systems use this interface to implement system objects and policies. In contrast,traditional operating systems are implemented with a fixed general purpose interface/abstraction to provide all features needed by all applications. This fixed high-level abstractions hurt application performance and flexibility because there is no single way to abstract physical resources or to implement an abstraction that is best for all applications. Exokernel operating system architecture approach tries to address this limitation of traditional operating systems by providing application level resource management.

The authors of the paper argue that since the applications knows better than operating system,what the goal of their resource management decisions should be and therefore, they should be given as much control as possible over those decisions. Thus they propose that the traditional abstractions be implemented entirely at application level. To provide the maximum opportunity for application-level resource management, the exokernel architecture consists of a thin exokernel veneer that multiplexes and exports physical resources securely through a set of low-level primitives. Library operating systems, which use the low-level exokernel interface, implement higher-level abstractions and can define special-purpose implementations that best meet the performance and functionality goals of applications. This structure allows the extension, specialization and even replacement of abstractions. Library operating systems can provide as much portability and compatibility as is desirable. Since the library operating systems are trusted,Exokernel separates protection from management through a low-level interface. In separating protection from management, an exokernel performs three important tasks: (1) tracking ownership of resources,(2) ensuring protection by guarding all resource usage or binding points, and (3) revoking access to resources. To achieve these tasks, an exokernel employs three techniques. First, using secure bindings, library operating systems can securely bind to machine resources. Second, visible revocation allows library operating systems to participate in a resource revocation protocol. Third, an abort protocol is used by an exokernel to break secure bindings of uncooperative library operating systems by force. Additionally by employing dynamic code generation performance of Exokernel improves.

The authors of the paper have implemented two software systems that follow the exokernel architecture: Aegis, an exokernel, and ExOS, a library operating system. The authors compare operation and performance of Aegis with Ultrix's basic primitives. Aegis represents the CPU as a linear vector, where each element corresponds to a time slice. Scheduling is done "round robin" by cycling through the vector of time slices. Aegis uses user-specified interrupt handling and exception handling code. Comparatively the performance of Aegis is faster than Ultrix. The ExOS manages fundamental operating system abstractions (e.g., virtual memory and process) at application level, completely within the address space of the application that is using it. ExOS operates efficiently in spite of executing at application level in part because the cost of crossing between kernel and user space is extremely low. ExOS is extensible and results show that extensions can have dramatic performance benefits. Different versions of ExOS can co-exist on the same machine and are fully protected by Aegis.

Shashi Shivaraju
XID:C88650674

The exokernel approach improves performance but also increases the complexity of the application/library OS,since library-OS needs to implement various system functionality. Thus it is a trade-off. The paper describes a prototype system for their performance evaluation,but further study must be undertaken on various platforms/systems to further validate the claims presented.