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

Name: Yash Aggarwal | yagga004 | 862333037

Summary

The paper proposes the idea of exokernel, an Operating System Architecture. This architecture claims to be an extendable, customizable, efficient, faster, and smaller operating system while maintaining all the features and functionalities of the current state-of-the-art operating systems. The architecture works on the principle that a kernel should be as small as possible with as few abstractions as possible. Abstractions by kernel result in less freedom for applications as some of the assumptions are made by the OSwhich results in hardcoded functionality. The applications running should have a say in management of the allotted resources so they can be optimized per the application's requirements. Exokernel removes many high-level abstractions that modern operating systems do, giving the control back to applications or library OSes, and only keeps low-level primitives that concern multiplexing and security. This means that the applications can control what resources they require and how best they can optimize them without interference from the kernel. By separating the security from management, the kernel can also better manage the security, as allocating resources to applications will now be more straightforward. Only a resource allocation table is required to know which application owns the resource, and then the OS can easily grant and revoke access to a resource while not looking at how the resource is being used. This will not only give customizability to the application but also take away work done by the OS. This will reduce the size and, thus, computational costs resulting in better performance.

Strengths:

- The paper proposes that the kernel should be as small as possible, with most policies and mechanisms taken out of the kernel and given to applications. The smaller kernel can be stored in physical memory, requiring fewer application-kernel switches. This will also increase the performance and speed of operations due to limited operations and fewer switching. As the kernel has limited responsibilities, it can be fast, efficient, and effective.
- The Exokernel with removed abstractions is customizable as now the applications can decide better on how they use the resources. These customizations can be implemented by the applications themselves or by library OSes. This also implies that multiple applications can have different ways in which they manage resources and interact with the kernel resulting in applications choosing the best possible implementation as per requirement.
- The exokernel OS can securely multiplex the resources with low-level primitives and secure bindings and can revoke access to resources when it detects foul play. This would result in complete control over resources with the applications with OS able to revoke access in case it is required.

Weaknesses:

- Although Exokernel proposes a no-trust design system in which all applications and library OSes are untrusted, and resources are multiplexed with secure bindings, thus applications cannot change/look into other applications' resources, there still are some concerns regarding the security of data. As Exokernel will only check the resource at the time of allocation, there could be applications that can misuse the resource provided. As the resource is held by an application and Exokernel has no knowledge of what the resource is being used for, there could be a scenario where an application with continuous NOP instructions blocks a resource.
- Removing all the abstractions creates additional overhead for application developers to either implement the resource management themselves or trust some library OS. This could result in application developers trusting some third-party library OS which may or may not be an ideal scenario.

Comments:

- The paper is on the right track in that the OS should be customizable in terms of services and abstractions, but it has gone too far. OS should provide all the functionality with an option/interface to implement/tweak/change the implementations in case a change is required.