OP1: Exokernel in Disguise

1 Ideas from the exokernel

One of the main idea of Arrakis [2] is that the role of the kernel should be limited to setting up data channels. Once this setup phase is over, applications can directly access the hardware, without any intervention from the kernel. This is similar to the idea in the exokernel [1] that the kernel should only protect resources, not manage them.

As both kernels only controls access to hardware, applications can choose a library operating system providing higher level abstractions, or create their own. Library operating system is also a concept proposed by the exokernel OS. These are libraries executing in the same protection domain than applications. They provide abstractions usually provided by the OS, such as virtual memory or thread scheduling. Applications can use existing one or create their own library operating system adapted to their needs.

2 Differences with the exokernel

Both kernels separate authorization from protection when using secure bindings. When an application wants to use a resource, the kernel performs authorization checks at bind time. At access time the kernel or the hardware perform only protection checks.

Arrakis uses hardware secure bindings, while exokernel uses a combination of both hardware and software secure bindings. In Arrakis, the kernel only setup the secure bindings. Then applications can directly access virtualized hardware without any kernel intervention as the hardware performs protection checks. Exokernel uses a combination of hardware and software secure bindings when an application accesses virtual memory. Hardware secure bindings are used when the virtual memory mapping is present in the hardware TLB. If there is a TLB miss, then the kernel handles the secure binding in software. The exokernel also implements software secure bindings for network by allowing applications to download code into the kernel.

Arrakis uses invisible resource revocation, while exokernel uses visible resource revocation. In exokernel, when the kernel wants to reclaim a resource used by an application, the kernel explicitly asks the application to give back the resource. This is called visible revocation. In Arrakis, the kernel allocates and deallocates virtual resources as needed without application involvement . This is called invisible revocation.

References

[1] Dawson R Engler, M Frans Kaashoek, and James O'Toole Jr. Exokernel: An operating system architecture for application-level resource management. *ACM SIGOPS Operating Systems Review*, 29(5):251–266, 1995. [online].

[2] Simon Peter, Thomas Anderson, and Timothy Roscoe. Arrakis: The operating system as control plane. In *Proc. 11th USENIX Conf. Oper. Syst. Des. Implement*, volume 38, pages 44–47, 2013. [online].