Critique Week 1

Andres Calderon - SID:861243796

January 11, 2016

SPIN - An Extensible Microkernel for Application-specific Operating System Services (Bershad, 1995)

The paper recognizes the evolving environment of current computer systems, which is still true. Certainly, there are new needs for increasingly exigent applications and systems. Operative Systems should evolve accordingly in order to support those requirements. The authors proposes SPIN with the intention to address the requirements of the "coming generation of *resource-intense* applications". The main idea of SPIN is that specific services and resources should be loaded as code sequences and installed in the kernel at runtime. It will allow different implementations and alternatives of existing interfaces according to the application's needs.

SPIN tenet is adaptability. It supports that through an extensible *microkernel* where its application-specific components are called spindles. The advantage of use spindles is that it allows the customization of interfaces and implementations for just those required kernel services. It is worth to note that authors state: "By allowing applications to participate in the implementation of services, we permit them to make informed decisions about their resources requirements". That gives some degree of responsibility to developers taking into account that they will know much better which kind of requirements and services they should use.

SPIN deals with resource management through two-level of resource allocation: system and user allocators. The former manages global resources, such as CPUs or network interfaces, while the latter manages private resources previously acquired. Just the system allocator is able to reclaim resources back when they are needed. Also, it is interesting to note that although each application can implement its own allocator, they are free to use default policies.

Adaptability is the key principle in SPIN. The best way to achieve it is using an actual programing language that allows applications to define and install new interfaces at kernel level. The main concern around this is how to perform secure compilation without penalize performance. Type safety and well-define interfaces are proposed to ensure legal operations but more advanced techniques and compilation technology are needed to ensure good performance.

Similarly to exokernels, the principles of SPIN seems valid and appropriate, however its adoption is not as expected. Personally, I find difficult to achieve optimal performance by runtime compilation. Increasing research has been done around this topic, even today, and I think adaptability and performance is a trade-off quite difficult to balance. I think that just under a specific context, where flexibility is cardinal, runtime-compilation microkernels could be an option.