The Multikernel: A new OS architecture for scalable multicore systems

Aaryan Bhagat

Idea rests upon assuming that to efficiently write an OS for multi-core systems, treat the system as some sort of independent units connected to each other through some network channels (like a distributed system with a common global clock). Implementing only message passing algorithms and the interface software for the hardware isolates the whole remaining OS and makes it more extensible and secure. Processes are executed through dispatchers which are mapped to the cores.Some properties act as global states like that of memory, memory safeguarding is done using the concept of capabilities which the authors borrowed (like many other things). Rest memory management implemented is done by user level code.

Pros

- Very extensible as it is not monolith and moreover, the OS code is separated from the hardware bindings. I can see this concept being developed from the microkernel paper somewhat.
- Optimizable, since we have different kernels of the OS to run different kinds of apps and code for memory management is user level along with a hardware neutral approach, we can write highly optimized user level code for each application specifically.
- Separation of kernels also yields a sense of security as bad code in one kernel need not necessarily corrupt other kernels.
- Kind of an alternative approach to the cache coherence approach (an approach with the aim of increasing multi processor efficiency but it is growing expensive these days).
- Works best for scalable systems, the authors have repeatedly stressed this point
- Message passing framework works fine as compared to the shared memory model.

Cons

- Memory management using capabilities as accepted by authors performs worse or equal to the normal memory management codes in monolithic kernels such as windows. Also the code is complex.
- Memory management has to be done by user level process which is overall bad as most application developers usually do not have that insight and they just write a simple working software using some high level language.
- Need to make consistency in data because we are now treating it as a distributed system, extra support in writing code for replication and consistency as well as 2 phase commit implementation of messages.
- Higher level operating system not tested in this paper which the author themselves admit.

Summary

Overall this paper was quite good, they are able to demonstrate at least using some basic benchmarks that their prototype non monolithic kernel can do demonstrably well to the current state of the art OS (which are already optimized a lot). Their idea of a distributed design is very helpful for scaling as they can even use theoretical methods to make it robust to faults, analyze the performance etc.