Review #3

The Multikernel: A new OS architecture for scalable multicore systems

A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, A. Singhania
ACM SIGOPS 22nd symposium on Operating systems principles, 2009

Jean-Pierre Lozi

April 30, 2010

1 Problem

Commodity computer systems containing multiple cores and/or processors are becoming more and more common. Most mainstream OSes have historically been developed for monoprocessors or multiprocessors having a limited number of processing units, using a model that scaled well in this context (global structures with locks, communication *via* shared memory, etc.). This model however has difficulties scaling well with newer hardware due to contention. Given the increasing number of cores in modern architectures and the complexity of interconnections between processing units, computer systems resemble more and more medium-scale distributed systems: designing an OS like an actual distributed system could improve scalability.

Newer architectures are also getting increasingly diverse (memory hierarchies, instruction sets, interconnects, etc.). However, OSes have been statically optimized for the most common architectures at a low level for now. The authors argue that given the varying nature of workloads and the diversity of hardware designs in modern computer systems, this approach will not be efficient enough anymore. They explain that designing OSes like distributed computer systems allows them to adapt better to various architectures, just like network applications are able to dynamically adapt to architecturally diverse networks.

2 Solution

A Multikernel is a new type of operating system that aims to address the problems discussed above. Multikernels are multithreaded, with an instance of the OS running on each available core. Cores do not share global structures as in a traditional OS; instead, the OS state is replicated on each one of them. Cores do not communicate *via* shared memory, relying on cache coherence algorithms; instead, all communication is explicit and happens through asynchronous messaging. Multikernels are also mostly hardware-independent. The only two architecture-specific modules are the messaging transport mechanism and the interfaces to the hardware (CPUs/cores and devices).

The authors wrote an experimental version of a Multikernel named Barrelfish. In Barrelfish, a *CPU driver* and a *monitor* are bound to each core. CPU drivers handle system calls, schedule processes and threads on their cores and perform other low-level core-bound operations. Monitors are processes running on each core that coordinate the system-wide state via messages and encapsulate most of the higher-level functions that are usually found in a traditional kernel. Barrelfish uses a *knowledge database* containing information regarding the underlying hardware (found by polling and measurements) that it uses to optimize its communication scheme. This database can be used to select appropriate message transports for inter-core communication or to allow for NUMA-aware memory allocation, for instance.

The ability to send messages from one process to another within the same core or across cores is

provided to user applications. However, applications can also communicate through shared memory like in a regular OS.

3 Performance

The authors performed experiments to evaluate the performance of Barrelfish's basic capabilities. They focused on the OS' handling of concurrency, messaging, computation and I/Os.

They showed that the performance of their messaging facilities was comparable to L4's IPCs, which is rather good even though L4 is not a fully-fledged, widly used microkernel. Performing a TLB shootdown (i.e. invalidating pages in the Translation Lookaside Buffer) and therefore unmapping pages was much faster on Barrelfish than on traditional OSes when the underlying hardware used more than 14 cores on their test configuration, showing Barrelfish's greater scalability in this context. Their experiments also showed that Barrelfish is faster than Linux for sending/receiving messages through the IP loopback (these tasks involve the messaging, buffering and networking subsystems of the OS). More experiments showed that Barrelfish has a comparable performance to that of Linux for compute-bound and IO workloads.

4 Benefits

Even though Barrelfish is an experimental OS, its performance rivals that of classical, highly-optimized OSes on computer systems that have a large number of cores, which tends to show that the Multikernel approach effectively delivers in terms of scalability. Traditional OSes may have not been optimized for these systems yet though, so we cannot be sure they will not scale well.

Given the maturity of mainstream OSes, rewriting one from scratch could be a daunting task, but Multikernels are fairly minimal (like microkernels or exokernels). Moreover, even though Multikernels are very different from traditional OSes on an architectural level, legacy applications can still run on them using traditional IPCs: user applications would not have to be redeveloped for a multikernel to become mainstream.

Therefore, if Multikernels effectively prove much more scalable than traditional kernels on the long run, they could overcome them in the future.

5 Shortcomings

Since all communication between cores is explicit in Multikernels, their development could prove extremely complex, especially since programming with asynchronous messages requires an event-driven approach. For Multikernels to be successful, they would have to prove much more scalable than traditional OSes, which is not clear: for instance, efficient algorithms would be needed to optimize communication between cores, and those used in traditional distributed systems may not be applicable.

Also, Multikernels are, in many respects, similar to microkernels, in particular regarding the copious amount of message passing required between user-space components. This caused a performance impediment which hindered the development of microkernels. The same problem could prevent Multikernels from being more than experimental proofs of concept.