The Multikernel: A new OS architecture for scalable multicore systems

Vidit Jain (NetID: vjain018, SID:862394973)

Summary

The Multikernel architecture proposed by the authors is prompted by several trends at the time in the hardware and OS landscape. Firstly, the prevalence of non-uniform hardware requiring tedious tailored optimizations in the OS. Secondly, increasing heterogeneity in cores within a single machine. And finally, the increasing feasibility of leveraging explicit message passing instead of implicit communication using shared memory. All these trends motivated the authors to develop an infrastructure where each core can operate in a more standalone fashion and the OS essentially acts as a distributed system. To achieve this, the authors have developed Barrelfish OS.

Strengths

Barrelfish OS essentially runs a separate kernel for each CPU core, negating the need for fine grained optimization for various hardware designs. To avoid running into scaling issues caused by cache coherence, Barrelfish uses explicit message passing system. Additionally, the machine state is maintained in replicate data structures allowing cores to process on locally available data, reducing latency. All these design choices allow the machine to scale effectively without running into memory bottlenecks, as evidenced in the evaluation section of the article. Low level operations such as virtual memory management are performed by the user-level code. However, the CPU driver is responsible for ensuring correctness of memory operations through retype and revoke. One key feature of Barrelfish is that the authors have allowed for flexibility if desired by the user. User applications can share capabilities and address space across cores.

Weaknesses

While the authors have provided significant details on the performance of key elements in Barrelfish infrastructure, it is still lacking on some important aspects. Notably, the authors did not discuss the implementation for maintaining replicated data structures and the performance impact of the same. In the same vein, the authors did not shed light on additional latency caused by CPU driver protection checks. Also, they have not discussed cases where it might be advantageous to not operate the OS like a distributed system, such as significant load imbalance amongst CPU cores. Finally, I feel that a performance comparison on homogenous CPU hardware would have been helpful for a more comprehensive analysis.

Other Comments

The authors recognized the future trends in hardware and OS design and presented a very innovate approach in the form of the Multikernel architecture. I found the evaluation section a bit lacking, however, I found the discussion around various design choices extremely comprehensive and intuitive. Also, I liked the additional section around potential future work detailing the edge cases such as sensitivity to message queue size potential and performance on machines optimized for sharing among cores.

Overall, I found the article extremely innovative, intuitive and timely.