Review, The Multikernel: A new OS architecture for scalable multicore systems

Robert Hoff

October 18, 2011

1 Paper Summary

The paper argues that because of trends in hardware diversity and core-count, architectural adaptions of operating systems are necessary. In particular, contemporary shared-memory systems place a theoretical maximum on the number of useful cores (some order of ten), but if Moore's law continues future machines may have hundreds or thousands of cores. [1] The proposed solution is to split the OS into multiple kernels, where each kernel acts as a mini-OS concerning it's own processor and independent memory. With such a system all IPC is performed by message-passing, but the system can theoretically scale to many more cores. The Barrelfish implementation, a prototype OS under development, is presented to test theory. Results indicate that Barrelfish performs better or comparable to existing systems on a selected sample of algorithms and applications.

2 The Problem

As modern CPU frequency scaling is prevented by physical constraints (overheating), the main focus towards performance is scaling the number of processors. The major concern in the paper is how efficient scaling in this dimension can be achieved to beyond just a few cores (multi-core) towards an unbound number (many-core). Processes on today's operating systems use the same virtual memory space, so even if processes can be allocated to different CPUs, the memory and cache coherence between cores need to be synchronised. The current method limits the number of useful cores that can be added to the system, because as the count grows, CPU cycles are wasted on processing contention, eventually gains in efficiency become zeroed out, and even inhibiting.

3 The Solution

The paper suggests rethinking OS design using ideas from distributed systems, the solution essentially is linking many machines on a single chip together to perform as one. The overall OS is split into several kernels, each acting as a mini-computer.

The processes operating within any of the given kernels are concerned with their independent memory space, reducing memory contention. IPC still being a necessity, is delegated to message-passing in the user-space. Barrelfish, a prototype OS to test the multikernel design, is under development but sophisticated enough to run selected tests against other existing systems.

4 Evaluation

Over a selected sample of tests Barrelfish performs comparable or better against existing systems. In the multikernel system, TLB shootdown is done by messages rather than inter-process interrupts (IPI). Above 14 cores, the preferred shootdown protocol performs and scales better than Linux and Windows IPI.

A collection of compute-bound workloads (algorithms) are performed against a Linux system, in all of these tests Barrelfish performs comparably to Linux. Finally in application tests, in UDP routing Barrelfish performs comparably to Linux. As a web and database server the system outperforms Linux.

5 My Opinion

A key concern, which is also highlighted in the paper, is if the tests performed are representative of real workloads. We can observed some research towards performance benchmarks, such as [2]

The specification in the paper include another main goal, that of core heterogeneity. In my opinion this doesn't seem relevant compared to the architectural rethinking required to take advantage of many-core scaling (as a generic problem).

It's observed [3] that present day operating systems which execute on multiple cores have evolved from uni-processor systems. Therefore we need to re-examine today's systems at fundamental levels. As number of cores grows, basic systems like scheduling and context switching have to relate to different performance criteria. Some responsibilities may be transferred to other sub-systems, for example that of ensuring high-priority threads execute first, may principally be a memory management responsiblity, rather than scheduling. I would agree with the authors that fundamental shifts in

operating systems architecture will probably occur to take full advantage of emerging trends.

6 Possible Questions

1. Why is core heterogeneity considered considered an issue, can't we build hardware to match our specifications? Do we expect better or different system performance running on diverse cores?

2. There is not much emphasis on which types of problems or algorithms we are expecting to compute. It would be interesting to know what the most important reasons are for increased 'commodity' processing power? What problems will we deal with in the future that we can't solve on commodity system today?

3. A question borrowed from [4]. How much we would be forced to rewrite our applications, to what extent can standard library interfaces be provided (C/C++)?

References

[1] Borkar S: *Thousand core chips: a technology perspective*, Proceedings of the 44th Annual Design Automation Conference (2007)

[2] Kuz I, et al: *Multicore OS benchmarks: we can do better*, Proceedings of the 13th USENIX conference on Hot topics in operating systems (2011)

[3] Wentzlaff D, et al: *Factored operating systems (fos): the case for a scalable operating system for multicores*, ACM SIGOPS (2009)

[4] Harris T, et al: *AC: Composable Asynchronous IO for Native Languages*, (2011)