Multicore OSes: Looking Forward from 1991, er, 2011

David A. Holland and Margo I. Seltze (Harvard University)

18.01.2012, Presentation By Paweł Hajdan

- software is not parallelized enough to take advantage of the more and more parallelized hardware
- how about moving to a shared-nothing architecture which is easier to scale?

But we've been there before!

- supercomputers of '80s and '90s hit similar problems
- shared-memory replaced by shared-nothing + message passing
- Map-Reduce is shared-nothing and hugely successful
- don't take the analogy too far

Make messaging really cheap

- support messaging natively
- cost comparable to a procedure call

Make messaging really cheap

- not just applications, operating systems too
- procedure calls are a special case of messages
- microkernel, “macrokernel”, or something completely different

A Model of Messages and Channels

- channel <- value (send)
- value <- channel (receive)
- channels can be sent though channels
- communication and also synchronization

Multiplexing

```
choose {
  option r1 <- c1: action1(r1); break;
  option r2 <- c2: action2(r2); break;
  option r3 <- c3: action3(r3); break;
};
```

Peer vs. Hierarchical Structure

- caller/callee relationship is asymmetric
- this leads to a hierarchy
- which can become cumbersome or wasteful

Implementing it

- UNIX and C
- needed in mainstream, must run legacy code easily
- concurrent dialect of C, lightweight parallelism
- more details are beyond the scope

Implementing it

- no kernel-mode/user-mode transitions
- no signals for asynchronous kernel -> app messaging
- designated kernel and application cores
- no need for special hardware support, for now

Implementing it

- the whole kernel must be architected to use message passing
- give every device driver its own thread
- less synchronization, cool!
- better fits static analysis models

Implementation issues

- most can only be discovered during actual implementation
- implementing choice effectively
- decentralizing virtual memory management
- too much parallelism and no middle ground

Implementation issues

- recovering from various failure scenarios
- scheduling threads on CPU cores
- scaling still not automatic, just easier

Conclusions

- lightweight shared-nothing message-passing parallelism
- alternatives like hundreds of VMs are unsatisfying
- is systems research community ready for the challenge?

Further reading

- BAUMANN, A., ET AL. The Multikernel: A new OS architecture for scalable multicore systems. In Proc. of the 22nd SOSP (October 2009). *(Barrelfish, Microsoft Research, Cambridge)*
- CLARK, J. Intel: Why a 1,000-core chip is feasible (December 2010). *(48 and 80 cores already there; how do we program 1000 core chip?)*
- GOOGLE. The Go programming language. http://golang.org/
- Jochen Liedtke's papers (microkernels, L4) (1991 - 2001)
- Pierre-Evariste Dagan. Language Support for Reliable Operating Systems (June 2009).

THE END