Multiprocessor Operating Systems

CS 6410: Advanced Systems

Kai Mast

Department of Computer Science
Cornell University

September 4, 2014

Let us recall

Multiprocessor vs. Multicore

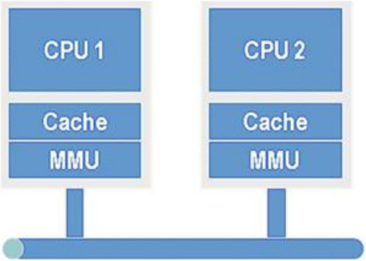


Figure: Multiprocessor [10]

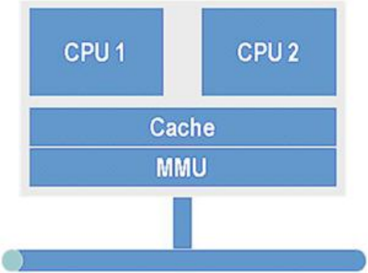


Figure: Multicore [10]

Let us recall

Message Passing vs. Shared Memory

Shared Memory

- Threads/Processes access the same memory region
- Communication via changes in variables
- Often easier to implement

Message Passing

- Threads/Processes don't have shared memory
- Communication via messages/events
- Easier to distribute between different processors
- More robust than shared memory

Let us recall
Miscellaneous

Cornell University

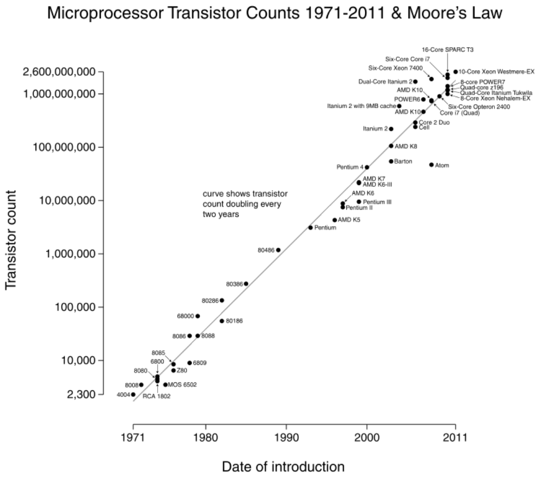
- Cache Coherence
- Inter-Process Communication
- Remote-Procedure Call
- Preemptive vs. cooperative Multitasking
- Non-uniform memory access (NUMA)

Current Systems are Diverse

- Different Architectures (x86, ARM, ...)
- Different Scales (Desktop, Server, Embedded, Mobile ...)
- Different Processors (GPU, CPU, ASIC ...)
- Multiple Cores and/or Multiple Processors
- Multiple Operating Systems on a System (Firmware, Microkernels ...)

How about the Future?

Moore's Law



(Source: Wikimedia Commons)

How about the Future?
Single-Core doesn't scale anymore

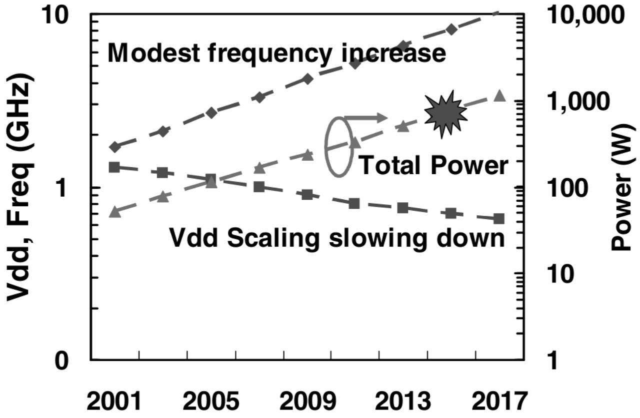


Figure: Possible power-consumption of a 10GHz chip [3]

How about the Future?

Rock's Law

- Manufacturing cost increases with amount of semiconductors
- Rock's Law eventually *collides* with Moore's Law
- One solution: Higher production quantity
- Another approach: Multiple mid-range processors instead of one high-end processor

How about the Future?

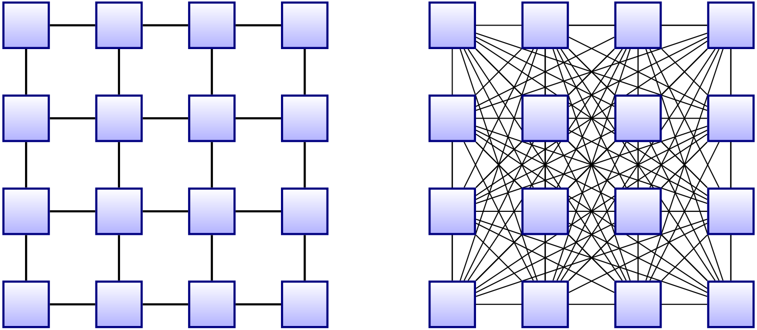
But...

Cornell University

- Multiprocessor Systems are reality today!
- Existing Operating System had to be adapted to support multiple cores
- Applications heavily rely on multi-threading (just think of the assignment...)

Interconnects are evolving

Direct Wiring does not scale



On-chip networks are more efficient in terms of power-consumption and area [2].

Interconnects are evolving
Many-Core Chips

Cornell University

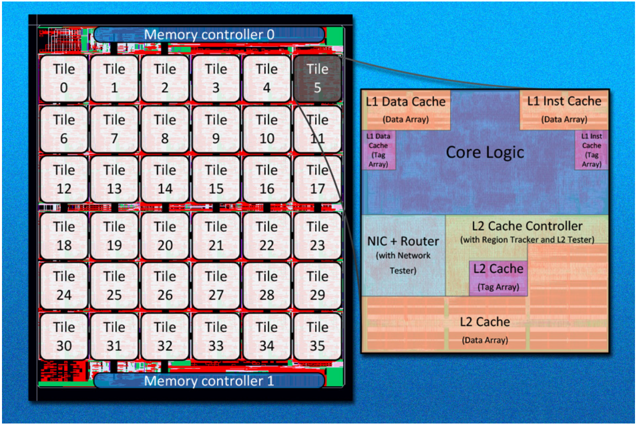


Figure: 36-core Chip from MIT [4]

Are Operating Systems ready for this?

In-Kernel Locking

Cornell University

n threads on n cores execute the following:

```
f = open("filename");

while (true) {
 f2 = dup(f);
 close (f2);
}
```

Are Operating Systems ready for this?

In-Kernel Locking

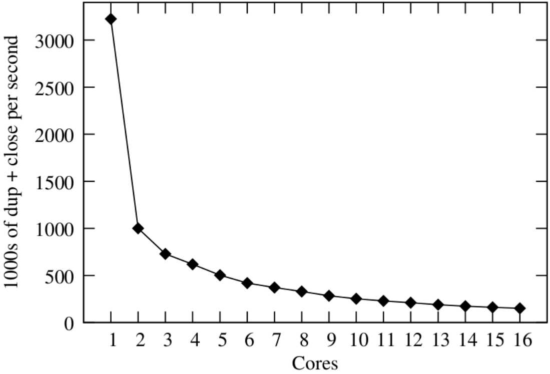


Figure: Decreasing performance with increasing amount of Cores in Linux [8]

Are Operating Systems ready for this?

- OSes optimized for most common configuration(s)
- Evolutionary improvements towards scalability
- Some special applications are highly coupled to hardware configuration
- Can we abstract from hardware **and** gain performance?

Multikernel and Tornado

Cornell University

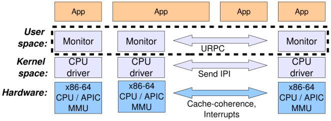


Figure: Barrelfish/Mulitkernel [1]

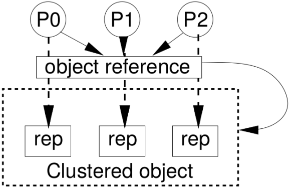


Figure: Tornado [6]

The Multikernel OS

The Paper

"The Multikernel: A new OS architecture for scalable multicore systems"

- Presented on SOSP in 2009

The Multikernel OS

Author Info

Cornell University

Andrew Baumann

- Was post-doc at ETH Zurich
- Now at Microsoft Research
- Several Projects focused around OS design

Simon Peter

- Was post-doc at ETH Zurich
- Now at University of Washington
- Current Project: Arrakis[9] (a Barrelfish fork)

The Multikernel OS

- The OS itself is a distributed system
- Actually, *multiple operating systems*
- *Explicit* communication between cores
- Abstract design to allow easier portability
- Note, that only the communication layer is abstracted

Barrelfish

What is it?

- Multikernel OS is just a concept
- Barrelfish is an example for an actual implementation
- Claims to have all the properties described before (scalable, modular, portable...)
- Let us evaluate and discuss later!

Barrelfish

Overview

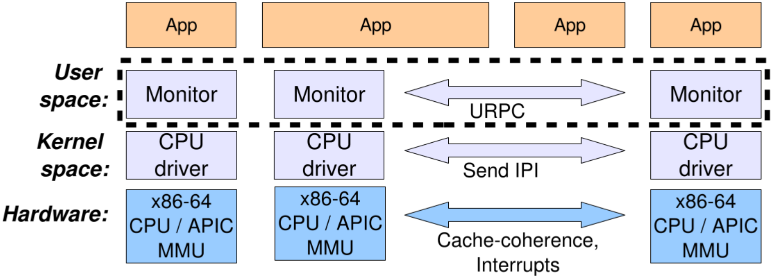


Figure: Structure of Barrelfish [1]

Barrelfish

Component Summary

Application

- (Possibly) distributed over several kernels

Monitor

- Generic (same for all cores)
- But still single threaded

CPU driver

- Architecture/Hardware specific
- Single-threaded

Barrelfish

Memory

- Memory is still a shared and global resource
- Logic is handled by the monitor, not the CPU driver
- Pages of memory a mapped to specific monitors
- But virtual/shared memory pages are also possible

Barrelfish

Performance Evaluation

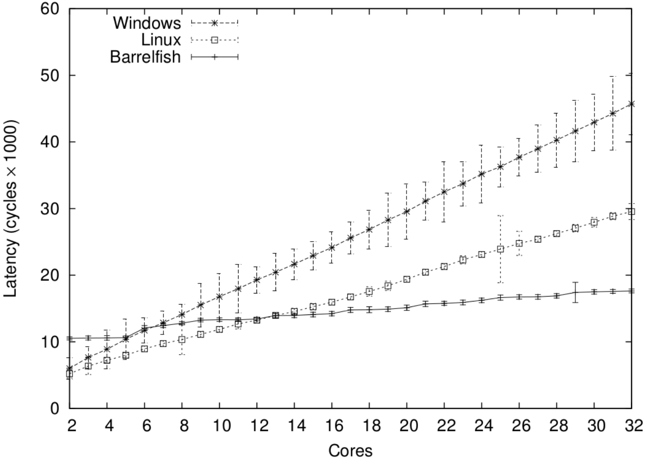


Figure: Latency of Unmapping a Memory Page [1]

Barrelfish

Performance Evaluation

Are the numbers meaningful?

- No complex applications were evaluated
- Only implemented on x86
- OS doesn't support any advanced features yet

Is this an important paper?

Pros

- Proposes a new type of Operating Systems
- The concept could represent a paradigm-shift
- Such an approach would make OSes ”future proof”

Cons

- No complex benchmarks exist yet
- Does not support systems that are distributed over the network

Open Questions

- Does it make sense to split monitor and CPU driver performance-wise?
- What would be a good communication model for Multikernels?
- How to support systems without a global shared memory?

Other Multikernels

Invasive Computing

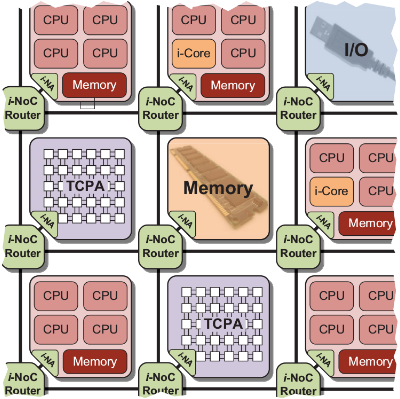


Figure: invasIC Architecture [7]

Tornado

"Tornado: Maximizing Locality and Concurrency in a Shared Memory Multiprocessor Operating System"

- Presented on SOSP in 1999
- Evaluated mostly on NUMAchine at UofToronto

Tornado
Authors

Ben Gamsa

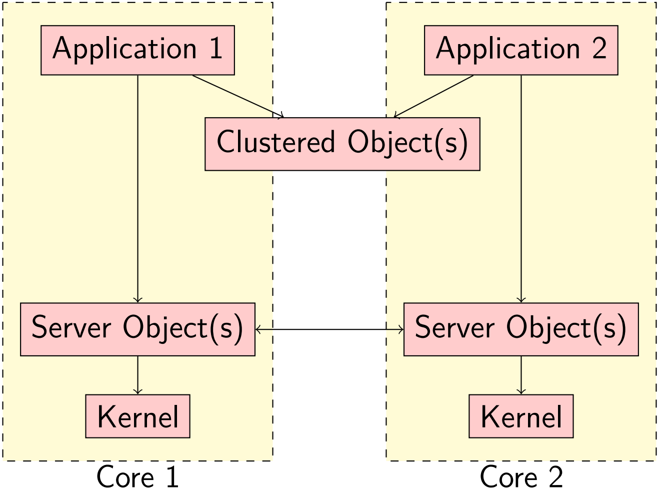
- Former Ph.D. student at University of Toronto
- Now working at Altera (unrelated to his research)

Orran Krieger

- Former VMware employee
- Working IBM T.J. Watson Research Center at the time of publication
- Now leading the "Center for Cloud Innovation" at Boston University

Tornado

Overview



Clustered Objects

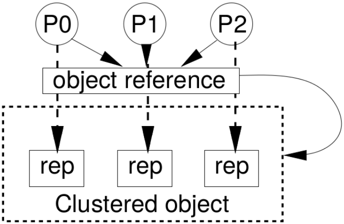
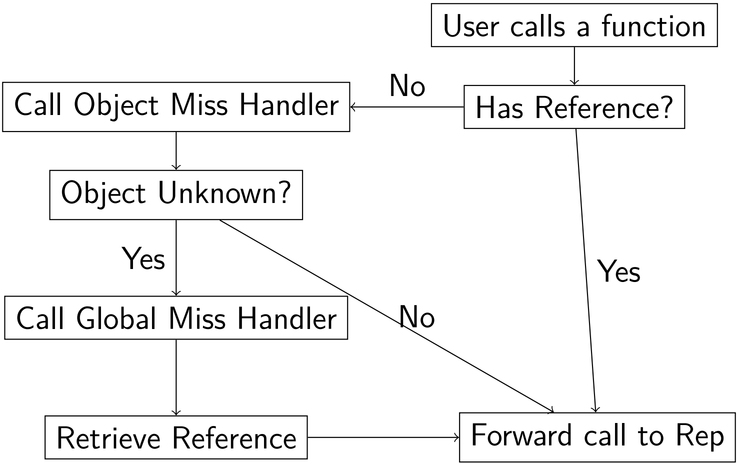


Figure: Tornado [6]

- Same problem as before: Some resources need to be shared
- Shared object can have more than one instance (or *representative*)

Resolving Clustered Objects



Resolving Clustered Objects

Miss Handler

Cornell University

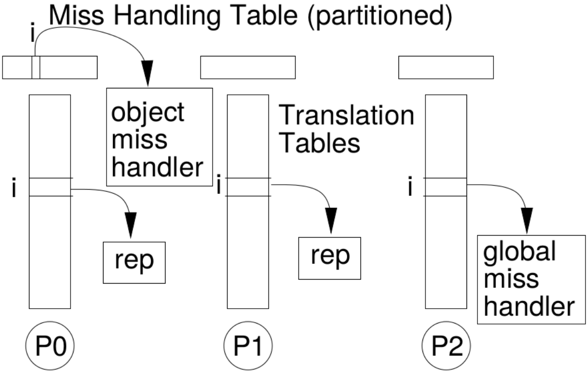
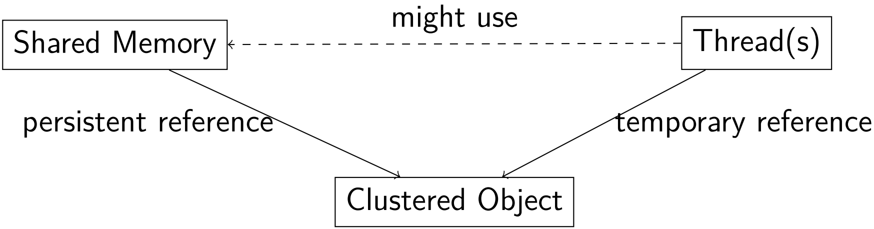


Figure: Miss Handling Table [6]

Garbage Collection



- Object must ensure that all references are gone before removal
- Fortunately, we know of all references because of the miss handler

Inter-Process Communication

Cornell University

- IPC is a core component of any modern OS
- Executing on local core is more effective (*handoff scheduling*)
- Cross-process call through local rep

Both Papers in Numbers

	Tornado	Multikernel
Authors	4	9
Year	1999	2009
Citations	182	497

Why does Multikernel seem to have a higher impact?

Conclusion

Similarities

- Threat OS as network of (almost) independent cores
- As little globally shared data as possible
- However, both assume global shared memory

Differences

- Tornado hides more from the user
- Barrelfish is built more modular
- Targeting different hardware (10 years difference)

Discussion

- Is the support for virtual memory a good idea? Should a modern OS expect the applications to do message passing?
- Is a hardware-neutral operating system realistic?
- Even with modularity, can one OS (architecture) cover all possible configurations? What about low-power embedded systems?
- Are the approaches really future-proof? What about systems that are distributed across the network?

Exokernels

Yet Another Approach

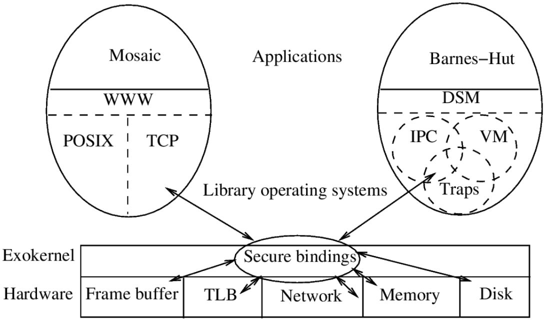


Figure: "End-to-End" Design of an Exokernel [5]

Exokernels

Corey

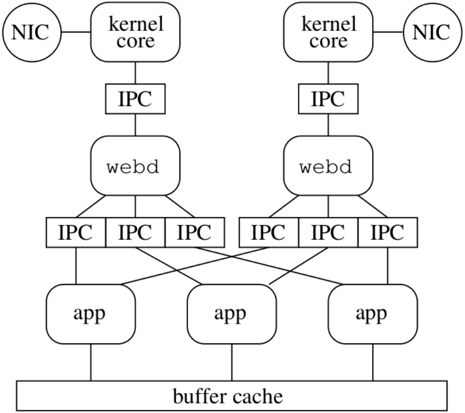


Figure: A Webserver powered by the Corey OS [8]

Exokernels

Arrakis

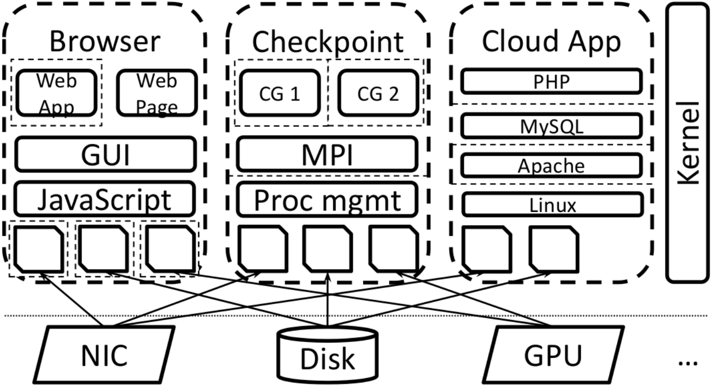


Figure: Design of Arrakis (a Barrelfish fork) [8]

References 0
Slides

- "Multiprocessors/Multicores"
  CS 6410 (Fall 2013) by Yue Gao
- "Operating Systems in a Multicore World"
  CS 6410 (Fall 2012) by Colin Ponce

References I

Literature

[1] Andrew Baumann et al. "The Multikernel: A New OS Architecture for Scalable Multicore Systems". In: *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*. SOSP '09. Big Sky, Montana, USA: ACM, 2009, pp. 29–44. ISBN: 978-1-60558-752-3. DOI: 10.1145/1629575.1629579. URL: http://doi.acm.org/10.1145/1629575.1629579.

[2] Evgeny Bolotin et al. "Cost Considerations in Network on Chip". In: *Integration-The VLSI Journal, special issue on Network on Chip, Volume 38, Issue* 38 (2004), pp. 105–128.

References II

Literature

Cornell University

[3] Shekhar Borkar. "Thousand Core Chips: A Technology Perspective". In: *Proceedings of the 44th Annual Design Automation Conference*. DAC '07. San Diego, California: ACM, 2007, pp. 746–749. ISBN: 978-1-59593-627-1. DOI: 10.1145/1278480.1278667. URL: http://doi.acm.org/10.1145/1278480.1278667.

[4] B.K. Daya et al. "SCORPIO: A 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering". In: *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*. 2014, pp. 25–36. DOI: 10.1109/ISCA.2014.6853232.

References III

Literature

Cornell University

[5] Dawson R. Engler, M. Frans Kaashoek, and James O'toole. "Exokernel: An Operating System Architecture for Application-Level Resource Management". In: 1995, pp. 251–266.

[6] Benjamin Gamsa and Benjamin Gamsa. "Tornado: Maximizing Locality and Concurrency in a Shared-Memory Multiprocessor Operating System". In: *In Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI*. 1999, pp. 87–100.

References IV

Literature

[7] Jan Heisswolf et al. “The Invasive Network on Chip - A Multi-Objective Many-Core Communication Infrastructure”. In: *Architecture of Computing Systems (ARCS), 2014 27th International Conference on*. 2014, pp. 1–8.

[8] Ong Mao et al. *Corey: an operating system for many cores*.

[9] Simon Peter and Thomas Anderson. “Arrakis: A Case for the End of the Empire”. In: *Presented as part of the 14th Workshop on Hot Topics in Operating Systems*. Santa Ana Pueblo, NM: USENIX, 2013. URL: https://www.usenix.org/conference/hotos13/arrakis-case-end-empire.

References V

Literature

[10] *Understanding Parallel Hardware: Multiprocessors, Hyperthreading, Dual-Core, Multicore and FPGAs*. URL: http://www.ni.com/white-paper/6097/en/.