POCS 2020
Exokernel Recitation

Rishabh Iyer

24.09.2020

Paper Recap

The idea (as in the paper)

- Operating Systems multiplex and <u>abstract</u> H/W resources
- No universally good abstraction that OS can provide
- Exterminate all OS abstractions

The idea (in POCS terms)

- Kernel and the OS should be distinct modules
- Kernel
  - Multiplexes resources securely
  - Privileged operations
- OS
  - Provide high-level hardware-agnostic abstractions
  - Can be unprivileged (e.g., libOS, microkernel servers)

Benefits (as in the paper)

- Efficiency
  - ❖ Applications can implement their own abstractions
- Reliability
  - ❖ Smaller OS implies fewer bugs
- Flexibility
  - ❖ Empowers users, easy to modify/implement new abstractions

Benefits (in POCS terms)

- Decomposing a monolith into modules improves
  - ❖ Flexibility and Efficiency
    - Can add/remove/replace individual modules as desired
  - ❖ Reliability
    - Faults isolated in smaller modules

Discussion

Exokernels, μKernels and VMs

- Apart from the exokernel, what other OS designs are you aware of?
- How do exokernels differ from
  - μKernels?
  - Virtual Machines?
- Today VMs are everywhere. Why not exokernels?

Enforcing resource modularity

- The exokernel enforces modularity using *secure bindings*
  - ❖ How does this work?
  - ❖ Applications can influence bindings by downloading code into the kernel. Does this break modularity?
  - ❖ Can you think of similar concepts used in other systems?

Talking PL

- Languages are moving to higher abstractions? Does this contradict exokernel principles?

Being precise about abstractions

- Can the CISC vs RISC debate be resolved using arguments similar to that in the exokernel paper?

Changing assumptions

- If designing a kernel for machines that run a single app, what changes would you make to the exokernel?

Will my app benefit from the exokernel?

- The paper has several examples of how porting an application to the exokernel and defining application-specific abstractions greatly improves performance. Can you think of applications for which this <u>will not</u> hold true?

Group design exercise

Pick one particular application and discuss how to redesign it to take advantage of the exokernel. Come up with concrete abstractions you would implement in your libOS to improve performance

Backup

Exokernel abstraction

- Is the exokernel the lowest level of abstraction an OS can provide? Can we go lower?
- Can we bake into HW a low-level of abstraction?