Singularity


JOURNEY

Language runtimes are slooooooow!

- Long-held objections to single-address-space systems:
  - Safe Language runtimes enforce a tax on all code
- “With garbage collection, the winning move is not to play.” -- some guy’s blog
  - You know it’s legit because “There are–and this is not a joke–over 100 citations in this blog post.”

Taxes of Hardware Protection

- Hardware isolation is not free --
  - Just turning on the TLB (paging) introduced ~5% degradation on a WebFiles benchmark
  - Separate aspaces increased cost to ~18%
  - Mode switch raises cost to ~33%
- We've lived with the cost for 30+ yrs;

What is verified?

- Type and memory safety of programs (via Sing#)
  - Corollary: SIPs are isolated from each other
- Adherence to channel contracts
- Correctly-versioned ABI usage

But in principle, anything? (via manifests/PCC)

Heap structure

- Processes have private heaps
- Exchange heap for transferring data cross-SIP
  - Exchange heap can only reference other exchange heap data
  - Exchange heap objects have at most one SIP owner
- So, zero-copy exchange between SIPs using channels

Garbage Collection!!!!!!!!!!!!!!!!!!!!!!!!

- In 2003, GC was critical to safe languages
- Singularity reflects that attitude
  - Every process has its own GC -- SIP isolation guarantees safety
- But today? Rust's ownership system is an alternative
  - Isolation helps again: can run both GCed and non-GCed SIPs without issue

Singularity kernel structure

- Single address space
- All ‘user’ processes & kernel run at CPL=0!
- All ‘user’ processes in C# / Sing#
- Process communication via typed channels
- Device communication via typed channels
- Drivers are run in SIPs

Singularity kernel implementation

- Mostly safe code
- Unsafe code required for GC, core MM
- Prototype work towards safe GC (otherwise GC can violate invariants)

Channels

- Processes communicate via messages on channels
- Contracts on channels, statically verified channel use
- SAS - ‘just move a pointer’
- Can we get channels without the rest of Singularity?

Hardware Implications

Does the design allow simplifying hardware?

Hardware support for verification?

“I would rather trust in the correct implementation of these mechanisms in hardware” -- systems grad student