seL4 Microkit
Ivan Velickovic
i.velickovic@unsw.edu.au
So, what is Microkit?

• An operating systems framework for building systems on seL4.
• Primary motivation is to lower the barrier the entry to developing on seL4.
• While making seL4 easier to use, we still want to uphold performance, security, and memory efficiency.
  • This means providing few, minimal, abstractions over seL4 primitives.
• Targeted at cyber-physical embedded systems, with a static architecture.
Why the name change?

• The “Core” in seL4 Core Platform makes it look like *the* only framework to build seL4 systems on which is **not** true.

• For those already referring to “seL4 Core Platform”:
  • `sed -i 's/seL4cp/microkit/g'`
  • Just kidding…

• If you spot anything not renamed yet that should be, please let us know! File an issue on GitHub or post on the mailing list.
Abstractions – Protection Domains

• An environment for executing user-level code.
• Single-threaded with its own address-space.
  • In seL4 terms, each PD contains its own CSpace, VSpace, and TCB.
• By default, all it can execute is its own code and *nothing* else.
• Execution is event-based.

PROTECTION DOMAIN

```plaintext
init()
notified(...)
protected(...)
```
Abstractions – Memory Regions

• MRs represent a contiguous block of physical memory.
  • Regular memory.
  • Device memory (for implementing device drivers).

• May be mapped into one or more PDs.
  • Allows for shared buffers between PDs.
  • Enables zero-copy communication.
  • Specify caching attributes and permissions.
Abstractions – Communication Channels

- Allows for bi-directional communication between a pair of PDs.
- Allows for synchronous and asynchronous communication.
- Notifications are used for asynchronous communication:
  - A PD “notifies” another to signal that some event has occurred.
  - Interrupts from hardware are also delivered as notifications.
- Protected Procedure Calls (PPC) are used for synchronous communication:
  - Enables a PD to execute code in a different PD.
  - For example, a client invoking some service in a server that returns a result.
Abstractions – Summary

CLIENT \hspace{2cm} Channel \hspace{2cm} DRIVER

seL4

Notification

MR access

IRQ

DEVICE
Microkit design

• All the PDs, MRs, and CCs are described in a System Description Format (SDF) written in XML.
  • This is deliberate, it allows trivial parsing as well as auto-generation.
• In addition to the SDF, the Microkit tool expects the ELFs of all PDs in the system.
  • It intentionally does not provide a build-system.
  • Each PD is linked with libmicrokit.
• Microkit is distributed as an SDK.
Status of Microkit

• After much discussion, the Microkit RFC has been approved by the seL4 Foundation.

• Microkit is now an official seL4 project: github.com/seL4/microkit!

• Development over the past year includes:
  • Limited dynamism (stopping, restarting, late-loading PDs).
    • Static architecture remains.
  • New abstraction - virtual machines.
  • Support for other architectures such as RISC-V and x86-64 and more hardware platforms.
  • CapDL integration to (eventually) connect Microkit to existing seL4 proofs.
    • An implemented verification story now exists.

• The process of upstreaming all the changes has started.

• You can follow the status of upstreaming here github.com/seL4/microkit/issues/61.
What’s next for Microkit?

• Development is most certainly not done!
• Enhancing the eco-system:
  • Virtual Machine Monitor (VMM)
  • Proper debugging support
  • Performance profiling
  • System visualisation tools
• Building a non-trivial example system using Microkit (PoS system).
Virtual machines on Microkit

• Why?
  • To avoid porting existing or implementing new device drivers for seL4.
  • Invoking legacy software.

• Main goals:
  • Secure and performant virtual machines.
  • Lower the barrier to entry for using virtual machines with seL4.
    • Having documentation and lots of examples is a priority!
Virtual machines – VMM as a library

• A "one-size fits all” VMM is not ideal.
• The library allows people to build their own VMM, with their own control-flow.
• Supports AArch64, RISC-V in-progress.
• Examples of using the VMM library in C, Zig, and Rust already exist.
  • Each of these is ~100-150 SLOC.
  • About 2300 SLOC involved to boot a Linux guest with libvmm.
Virtual machines – Pass-through devices

• The easiest way to get I/O in a virtual machine is “pass-through”.

• This gives the guest full access to a certain device.

• In Microkit, this is trivial to do by creating a memory region and mapping it into the virtual machine.
Virtual machines – Device sharing

• We need to be able to share devices.
• Using and extending the sDDF transport layer, we can allow native clients and other virtual machines to make use of the same device.
• sDDF allows us to transparently swap out a virtualised driver with a native driver.
• We are working towards graphics and networking support.
Virtual machines - Summary

• Sufficient for development and experimentation, but not production ready yet.
  • Proper performance and security analysis needed.
  • Highly-used features such as SMP guests and virtIO are still in-progress.
• Not just for Microkit! The project should be able to be used in other seL4 environments.
  • The library depends on few seL4 invocations.
Proper debugging

• When running on real-hardware, using only printf debugging is quite limiting.

• Adding GDB support to Microkit to provide the ability to:
  • set breakpoints (both in software and hardware).
  • single-step code.
  • inspect kernel state, such as dumping a CSpace.

• Also want to provide stack traces for faults, such a virtual memory fault.

• Mostly a work-in-progress at this stage.
A performance profiler

- Current profiling on seL4 is limited.
  - Good for getting an idea of cache misses, kernel entries etc.
  - For non-trivial systems, we need a more systematic way of tracking performance.

- Goal is to have a statistical sampling user-level profiler to track performance of each PD in the system.

- Allow analysis of data by existing tools such as perf.
- Export data over serial, network, block.

- One potential problem is that kernel changes are required, conflicting with simply attaching the profiler to a deployed and running system.

- Again, mostly a work-in-progress at this stage.
Community input

• As the main developers of Microkit, there are only so many use-cases we have considered.
  • This means we are bound to miss some use-cases and there may still be holes.

• While we do try to give users of Microkit the best user experience, there will almost certainly be gaps and mistakes as the project matures.
  • Ranging from documentation, to error messages, to workflow, etc.

• It is vital for the community using our software to tell us what needs improving!
Thanks! Questions?