Bridging Academia and Industry

seL4 Summit 2024

Sydney, AU

10/16/2024

Yanyan Shen

Dhammika Elkaduwe

Manjunatha Swamy

Overview

seL4

A high-assurance, high-performance operating system microkernel

SkyOS-M

A vehicle operating system requiring high dependability and performance.

- System servers
- Driver framework
- POSIX
- 3rd libraries
- Software development kit

- System / performance utilities
- System monitoring
- Comprehensive test suites
- Hard work and optimism

The Essentials

Servers

- A classic multi-server design: process (aka root server), time, device, network, and file.
- Properly layered servers: to avoid circular dependency. (A hang happened in an unexpected way …)
- Process server: managing processes, kernel objects, and service namespace.

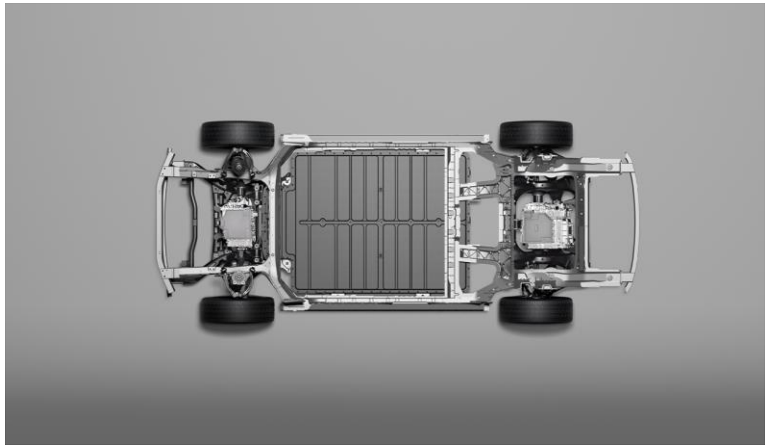
Core Libraries

- The interfaces provided by the servers.
- The interfaces used by native applications, servers, and drivers.
- Not POSIX, but good enough for building everything from scratch.

You do not have to be an seL4 expert to develop servers and drivers for SkyOS-M (well, eventually you will :-)

Drivers

- Each driver is still a separate process, not a shared library linked with a server.
- A device driver framework handles the common operations, and a driver just needs to implement cared functions.
- The device server starts drivers according to a device tree file and allocate MMIO regions and IRQs based on the file to drivers.



POSIX and 3rd Libraries

Common Questions

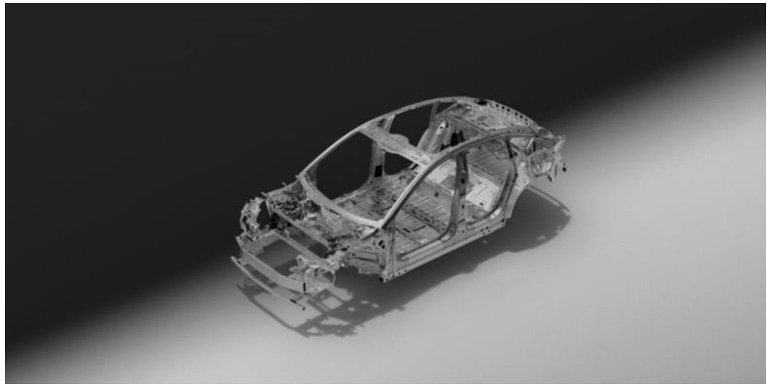
- Does it support fork?
- Does it support mmap? How about file-based mmap?
- Does it support select and poll?
- Does it support signals? Signals to threads?
- Even eventfd ...

Benefits

- Help the adoption of SkyOS-M by application teams.
- Enable software reuse.
- Build the foundation for a wider adoption.

Answers

- Implement the commonly-used POSIX APIs.
- Porting of musl libc by seL4 foundation is a good start.
- Libvsys is the layer emulating Linux system calls used by musl libc with core libraries.
- For instance, signal is purely emulated in user mode without kernel changes.



Tools

SDK

- Create applications without knowing about the underlying seL4 and OS framework.
- Use Conan recipes for managing dependencies.
- Manage prebuilt binary files to reduce build time.
- Generate disk images based on the recipe contents.
- Launch QEMU to run the whole OS with selected libraries and applications.
- Manage releases.

You do not have to be an OS hacker to develop applications for SkyOS-M.

Utilities

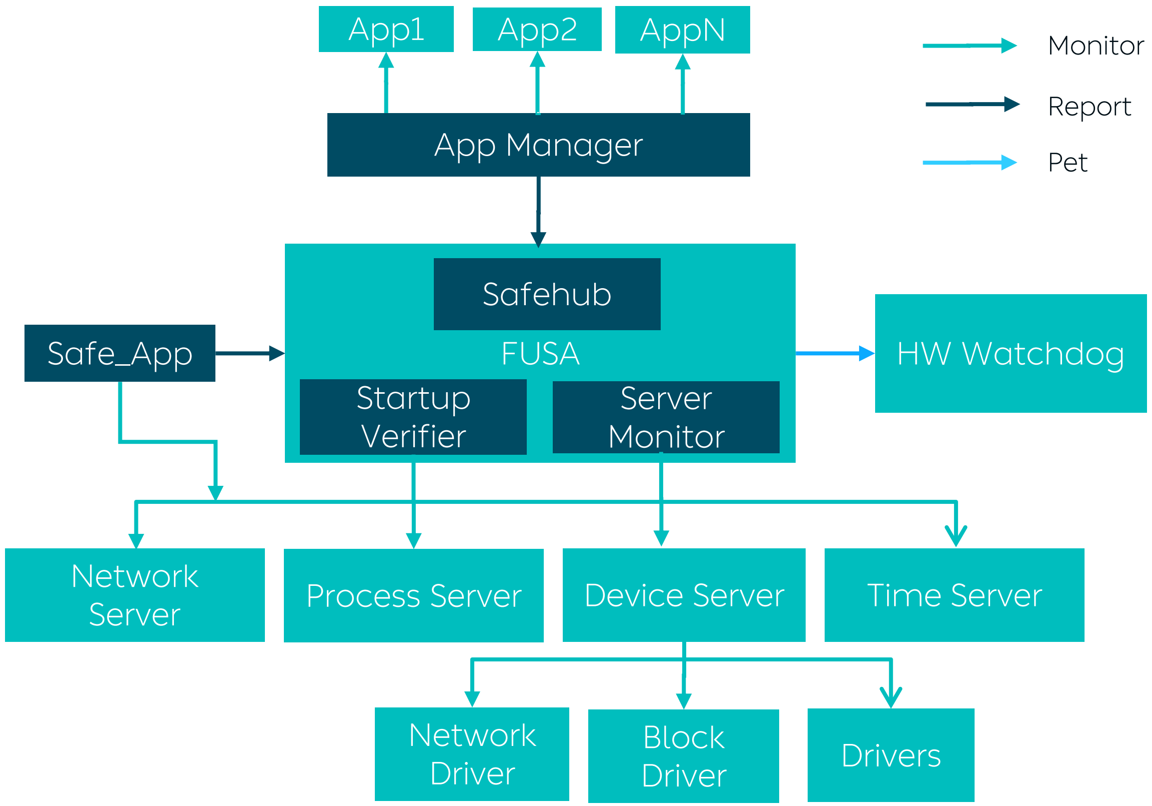
- Toybox – plenty of shell toys.
- Some tools (ps, top, lsof …) have to be built.

Debug Tools

- Kernel / application syscall tracing.
- System / application status dump.
- Automated crash backtrace file.
- Continuous system status monitoring.
- In-memory minimal system (when FS or EMMC hangs).
- GDB.

FUSA

Hierarchical Monitoring



No Formal Verification Yet

Levels of Tests

- Unit, component
- POSIX conformance (LTP)
- Scenarios
- Fault injection
- Stress (stress-ng)
- Performance
- System integration

Defects Discovered

- Around 3000 defects, various critical levels.
- Only 2 kernel bugs, of course the unverified parts (kernel SMP lock and IPI).

Snapshot on Tests

- Merge request tests: 4500 / 60 mins (QEMU only)
- Nightly tests: > 6000 / 300 mins (QEMU and HW)
- Stress test: > 400 / 60 mins & 12 hrs
- Performance tests: > 200 / 200 mins
- Conformance tests: 3000 / 70 mins
- Fault injection tests: 800 / 20 mins

Numbers are approximate.

What Do We Learn?

Building a proof of concept is easy, but productizing the PoC is, at least, 10 times harder.

- Some stress-ng stressors failed immediately after they started; some ran forever (hung). -> Tests with a set of mixed stressors (FS, network, VM, etc.) ran 12+ hours.
- An important scenario test had a failure rate of 40% initially. -> 1000 runs all passed.

Resource management is fundamental, but it is very difficult to get it right.

- A new resource leakage type: CSpace slot leakage.
- When a process terminates, server-side resources for the process must be freed.

The multi-server approach and POSIX do not work efficiently.

- File-based mmap needs multiple calls between process server and file server.
- The select/poll need to multiple calls to required servers.

Tooling is essential for OS developer productivity.

- How would you debug when you cannot run any commands when FS or EMMC fails?
- How would you debug when both network and console are down?

Delivery first.

- Working on a reasonable solution and searching for a better one.
- Make it work first, even if the architecture or code is not ideal.

Gain users first; improvements follow.

- Discover APIs use patterns not covered by tests.
- Help the users to tune their applications for the OS for improved performance.

Stay calm and trust your kernel!

What Would We Do Differently?

Shift the functionalities and bookkeeping data of servers to libraries linked with applications.

- Keep client states on servers as little as possible to reduce the complexity of server restart.

Prefer notifications to endpoints for I/O syscalls.

- Avoid the syscall restart issue when handling signals.
- Reduce message copy overheads.
- Support async IO naturally.

Support Rust early.

- Implement servers in Rust to reduce memory issues.

Use clang as the default compiler from the beginning.

- Better integration with libc++.
- LLVM-based safety compilers.

Tackle complex POSIX APIs early.

- Should just bite the bullet.
- Would have more time for improvements.

Make sure that every OS develop watched the AOS courses and read the seL4 manual.

- Just too many new concepts to digest.

Make a Wish

- Page map syscall supports multiple pages in one go.
- Reduce VSpace bookkeeping overhead (aka shadow page tables).
- Page-table-level share to avoid duplicating and mapping frames.
- A verified big-lock kernel.
- A clustered multi-kernel.
- A fine-grained locking kernel.
- More real-world systems to learn from.
- A toolset or approach for formally verifying core components mostly automatically.

Random Fun

- Call printf, malloc, or others, before the code finishes with contents in IPC buffer.
- Memory issues are still very challenging.
- If an issue can be reproduced on QEMU, it is probably a logic bug.
- When you suspect there a bug in the LibC / compiler / kernel, it is probably in your code.

NIO

Thank you