NIO

Towards Dependable System Services

seL4 Summit 2025

Prague, CZ

09/03/2025

Yanyan Shen

Dhammika Elkaduwe

Backgrounds

Faults

A flaw or defect in the system

- Intentional: attacks, viruses, malicious logic …
- Accidental: programming bugs, race conditions, transient hardware issues ..
- External: power outage, physical attack …

Errors

A manifestation of a fault

- Incorrect register values
- A busy looping thread
- Corrupted memory

Failures

A visible outcome of an error

- Program crashes
- System crashes
- Loss of data

Detect and Recover to Improve Service Availability

How Can seL4 Help

System Software Supports for FT Systems

Isolation between the kernel and system services
- When a service crashes, the kernel still runs.

Process isolation and fault containment
- Each service has its own address space.
- When the file system is down, the scheduler or network still runs.

Health checks and monitoring
- Exception handler
- Watchdog
- Integrity checks (data checksum, replication)
- Liveness

Recovery
- Process restart
- Checkpointing and rollback
- IPC retry/timeout/error report

A Verified Kernel

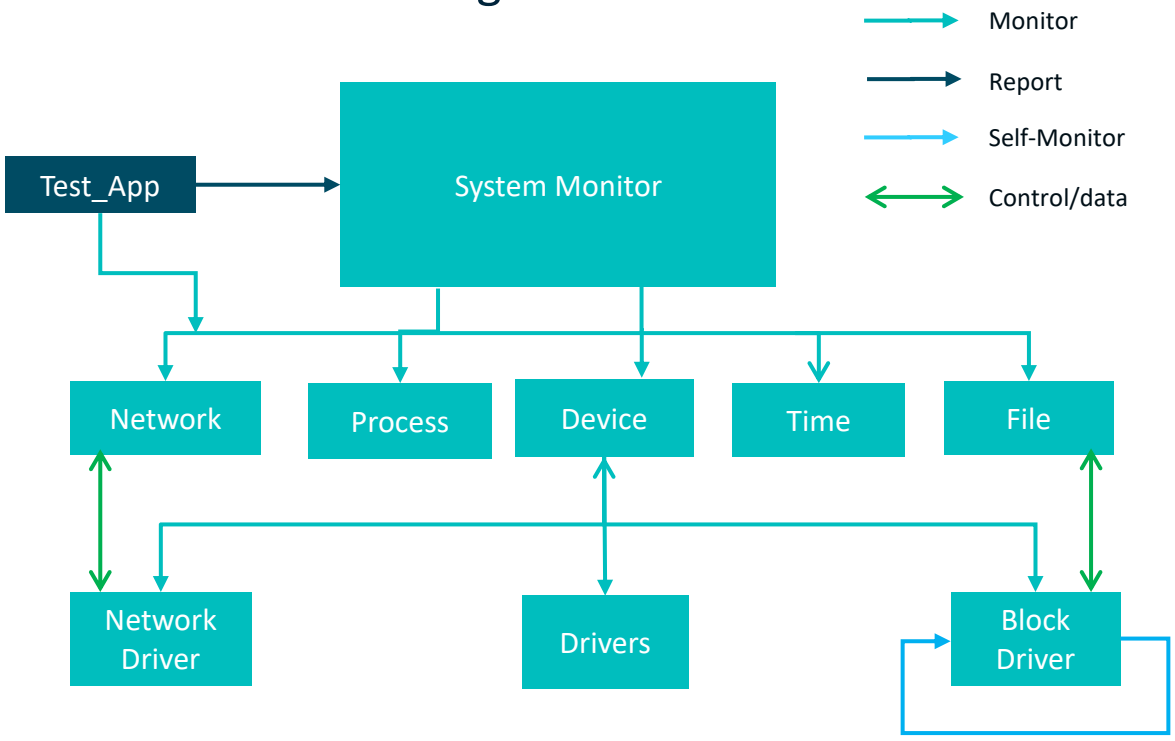
- No accidental programming bugs
- Small TCB in privileged mode
- Less total execution time in privileged mode

Microkernel Architecture

- User mode fault handling
- User-mode services: fault containment
- Micro restart
- Policy-free

System Health Monitor

Hierarchical Monitoring



- System monitor watches servers
- Device server watches drivers
- Direct fault handler: exceptions, cap faults
- An external test app to trigger operations
- Self-monitoring
- Check if a service is alive and working
- When system monitor or process server fails, a system reset is triggered.

Recovery Approaches and Challenges

Backward Recovery
- Checkpoint and rollback
- Journaling/logging

Forward Recovery
- Error masking (redundant execution, error-correcting code)
- Retry with alternate resource

Replication-Based Recovery
- Lock stepping
- Active-active, active-backup

Service Restart (our topic)
- Restart a faulty component without stopping the whole system and continue running

Detection Accuracy
- Timeouts of heartbeats should be tuned for services
- Avoid needless restarts caused by false positives
- Long service delay caused by false negatives
- Performance overhead

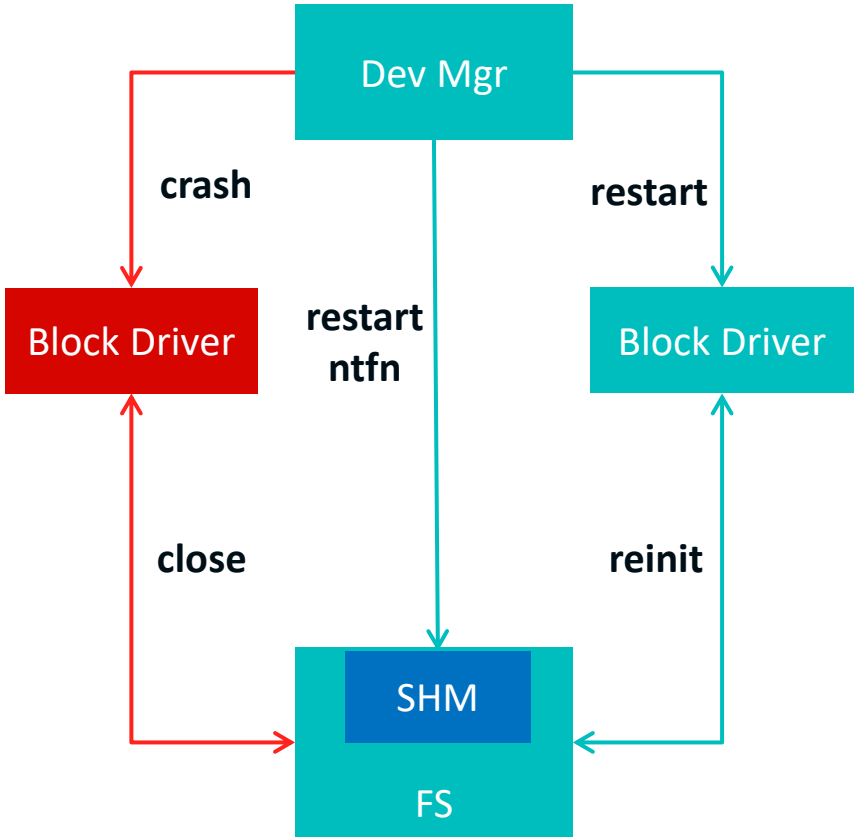
Transparent Restart?
- Ideally, transparent to applications
- Visible to other OS service components

State Management
- Most servers have states
- IDs or sessions maintained with clients
- In-flight requests

Complex Recovery Logic
- Bugs in recovery code
- Tests for uncommon code paths

Device Driver Restart

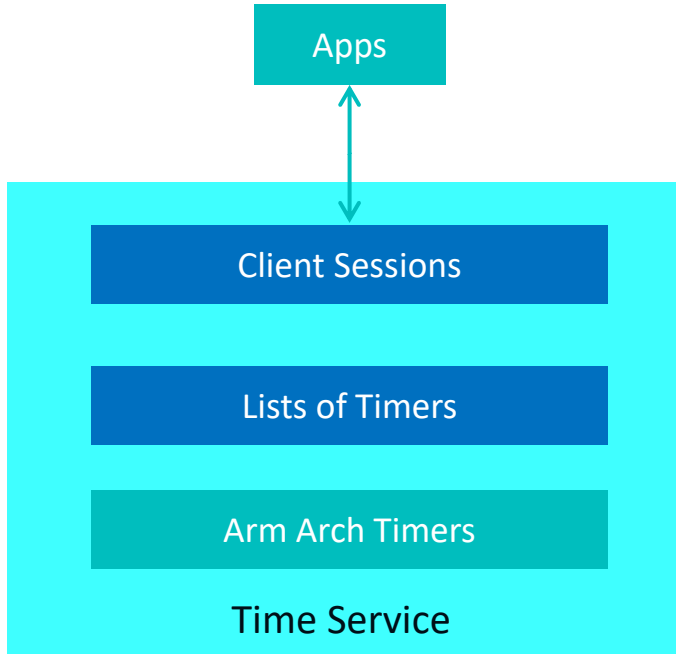
A block driver as an example

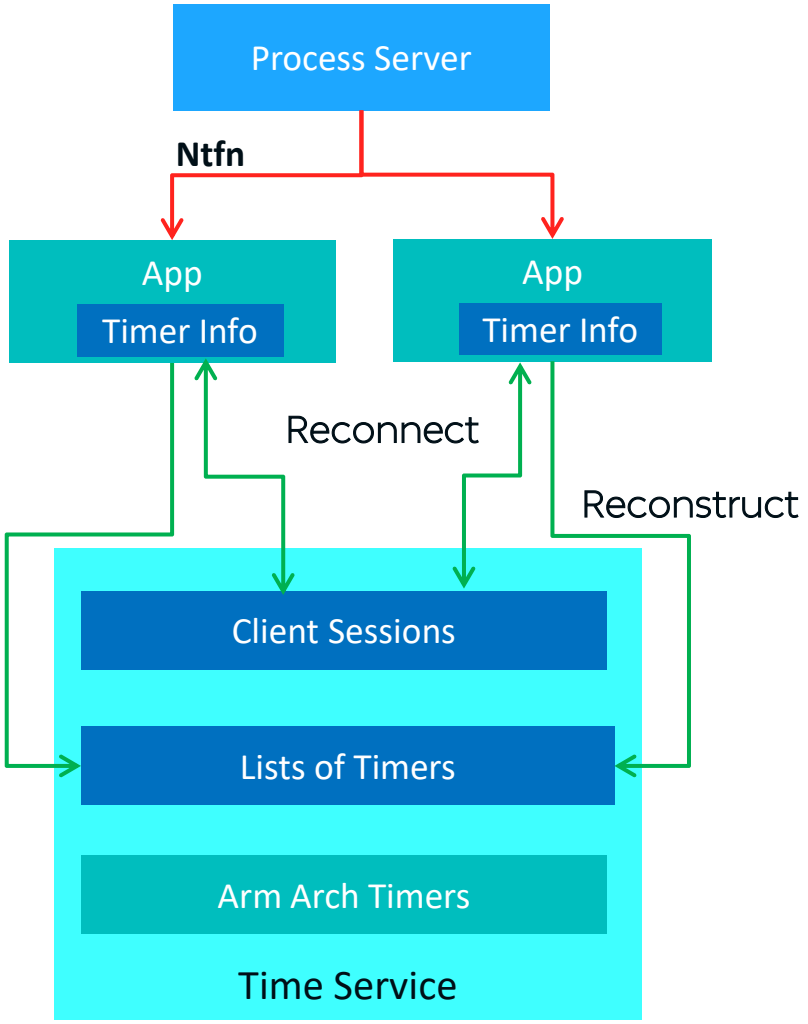


- Each driver is a separate process.
- IPC and shared memory are used for connecting the clients and drivers.
- ELF files of drivers are kept in memory.
- Requests and responses are in the shared memory region, which is preserved during driver restart.
- A server waits until a driver becomes ready after a restart and reestablishes control/data paths.
- Unfinished requests in the SHM will be retried by the restarted driver.
- Few internal states.
- The data integrity of SHM needs to be protected by other measures.

Server Restart - Time Service

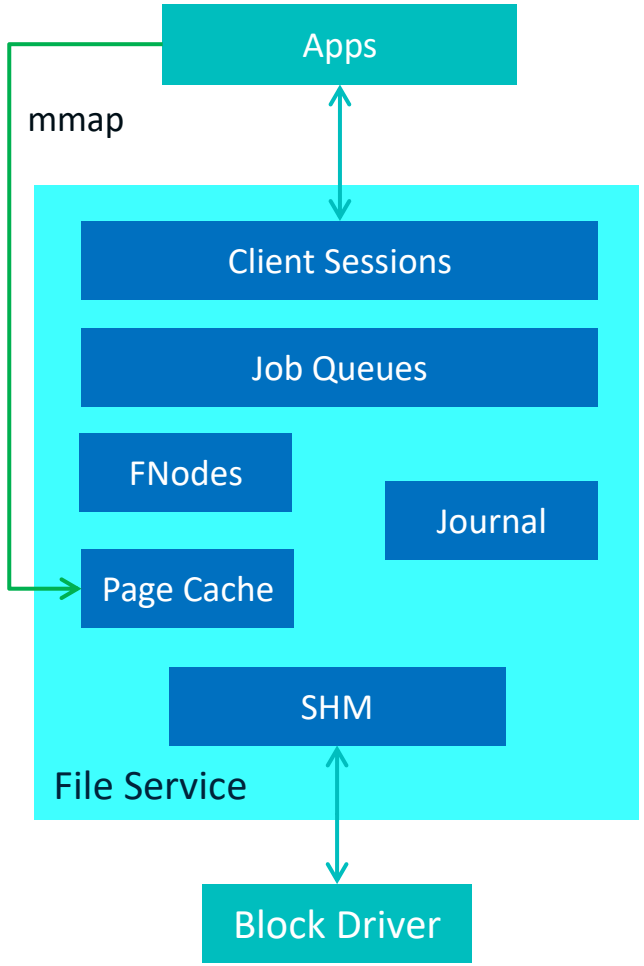
- Maintain wall clock
- Serve one-shot or periodical timers





- An app keeps track of timers it uses. This is hidden in the library so that the callers of the POSIX APIs do not need to modify code.
- The states of time services can be reconstructed by recovering timer information saved by the apps.
- A background thread reinitializes the timers.

Server Restart - File Service



Client Sessions
- A copy of open files and states are saved in the clients' data area.
- Sessions can be reestablished when clients detect issues.

Job Queues
- Pending jobs are discarded, clients will reissue.
- Interrupted jobs need to be examined individually.

FNodes
- Gradually reconstruct based on clients' data.

Page Cache
- Preserved across restart since apps still access it while restarting.
- Reconstruct the relations with FNodes from data from clients and process server.

Journal

SHM with Block Driver
- Preserved across restart.

Summary

- seL4 and microkernel architecture align perfectly with the architectural requirements of a fault-tolerant OS.
- Hierarchical monitoring is critical for timely detection.
- Service restart is a practical and promising approach to tolerating accidental faults.
- Stateless services are straight-forward to support restart.
- State management for existing complex services needs to be examined carefully and individually.

Future Work

- Shift more states to applications to simplify state management and service restart.
- More comprehensive fault injection campaigns to evaluate the effectiveness.
- Mechanisms to ensure that the states kept in applications are not modified by user code.


NIO ET9
敢 想 ， 有 为


THE ALL-NEW
NIO ES8
创 领 新 境


东道 L90

Thank you