Hochschule **RheinMain**
University of Applied Sciences
Wiesbaden Rüsselsheim

WAMOS
Benefits of dedicated hardware for microkernels

Date of last update: 24. August 2017

Daniel Schultz

Hochschule **RheinMain**
University of Applied Sciences

OUTLINE

1. Background

2. Memory allocator

3. Scheduling

4. Conclusion

BACKGROUND

MICROKERNELS

Microkernels are operating systems with less functionality in the kernel itself.

- → Interprocess Communication
- → Scheduling
- → Memory management

sel4 was written from scratch by the NICTA group with the aim of providing a basis for highly secure and reliable systems.

FIRMWARE

Linux covers many embedded system use cases and all semiconductor provide kernels for their hardware.

Firmwares have only one purpose; control the hardware and supply a interface to external components.

Firmware-controlled hardware is generally designed for one task.

MEMORY ALLOCATOR

BUDDY SYSTEM

Used by seL4 and Linux for memory management.

The buddy system is an algorithm to split one big piece of data with the size power of 2 into smaller fragments.

When all fragments are made of size power of 2 this can be represented as a binary tree.

BUDDY SYSTEM

Let there be data of 2^k words and a depth n of the binary tree.

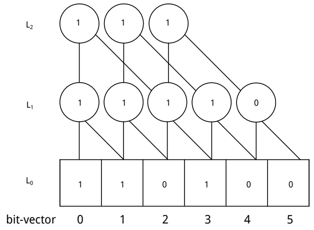
Each level h, with $0 \leq h \leq n$ has 2^h nodes and each node $T_{h,j}$ represents data of size 2^{k-h}.

All allocatable block sizes are $2^k, 2^{k-1}, ..., 2^{k-n}$ words.

For example, a tree with $k = 15$ and $n = 8$ describes data of 1 block with 32786 words (1 MB), 256 blocks with 128 words (4 kB).

Searching for data size of 2^{k-h} requires a look into level h of the binary tree.

HARDWARE



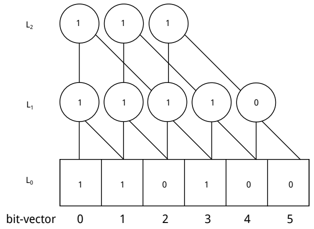
Each leaf node with size of 2^{k-n} will be represented by one bit.

The bit-vector has a size of N.

An OR-gate prefix logic decides, which data is free and requested.

Each OR-gate $T_{h,j}$ is connected with two gates at $T_{h-1,j}$ and $T_{h-1,j+2^{h-1}}$

HARDWARE ALLOCATION

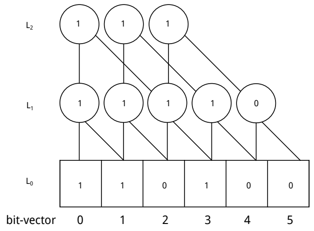


Searching for data size of 2^{k-h} at level L_h.

If a free node was found, all accessible bits get flipped to a 1.

The accelerator returns the physical memory address.

HARDWARE DEALLOCATION



Calculate the bits by a given address and the memory size.

Flip all these bits to 0.

USAGE

Only 5 cycles are needed to find free data or 50 ns if the hardware is driven by a 100 Mhz clock.

The smallest available block size, 2^{k-h}, is equal to a system page size.

The accelerator can handle the same size of memory like the RAM controller, but a barrier can shrink the OR-gate.

This hardware can handle page faults in a fast and efficient way.

SCHEDULING

CURRENT SCHEDULER

The current seL4 Kernel uses a priority-based round-robin scheduler with lists of all threads for one priority.

The scheduler will choose all threads in sequence from head to tail for each priority.

In a system with N threads, each thread will get $\frac{1}{N}$ of the global CPU time.

This procedure is fair but it is not real-time capable.

REAL-TIME SCHEDULER

Instead of the time-slice, a Scheduler Context (SC) was added to the Thread Control Block (TCB).

The SC contains information about the time budget and period a thread.

If a thread exceeds its time budget, it will added to a so-called release queue.

All exhaust threads are collected and ordered by the next budget refresh.

The scheduler can choose the highest thread from either the priority list or release queue.

REAL-TIME SCHEDULER

Introduction of passive server, which borrow time from the caller.

A thread can ask for more time to clean up or rollback data.

All event-driven threads also need available time budget, otherwise a pending interrupt occurs.

HARDWARE

The seL4 scheduler is optimized in software can not mapped directly in hardware. To reduce chip space it will be limited:

→ 8 priorities

→ 64 threads

→ Times greater or equal to 1 us

This reduced scope will still meet the scheduler criteria.

HARDWARE

A internal small RAM holds priority, TCB address, budget and period for all 64 threads. The remaining time and next timestamp will also saved as private data.

The selector will simply find the next thread by walking over each memory structure.

A watch dog will be armed with the budget time and triggers an interrupt to the CPU.

After the watch dog triggered, it sets a ”release queue” flag to the thread structure.

PROBLEMS

Round-robin scheduler in hardware are common in switches for a static count of tasks, but not in dynamic environments.

Changing the priorities of threads at run time is missing.

CONCLUSION

CONCLUSION

Interprocess communication still exists on many platforms.

The memory management accelerator can reduce page handling to a constant time of few nanoseconds.

The real-time scheduler finds the next threads and helps the kernel to save dynamic memory and calculation time, but it has a lot of potential for a smaller footprint.