THE UNIVERSITY *of* EDINBURGH

Adaptable Multicore Scheduling

Kimberley Stonehouse, PhD student

Background

- InfOS is a research operating system
  - Targets the 64-bit x86 architecture
  - Object-oriented and written in C++
  - Relatively small codebase (20,000 loc)
  - Feasible candidate for a redesign

Goal: redesign InfOS to support multicore processing!

Motivation

- Multicore architectures are ubiquitous
  - IBM led the way with the dual core POWER4 in 2001
  - Fugaku was the #1 on Top500 with 7,630,848 cores in 2021

- But multicore programming is **hard**
  - Introduces synchronisation difficulties
  - Locking primitives are a solution **if applied correctly**

Motivation

- Established operating systems are open-source, but **complex**
  - The Linux kernel is written in C in a semi-object oriented style
  - Codebase evolved over 30 years and has 27.8 million loc

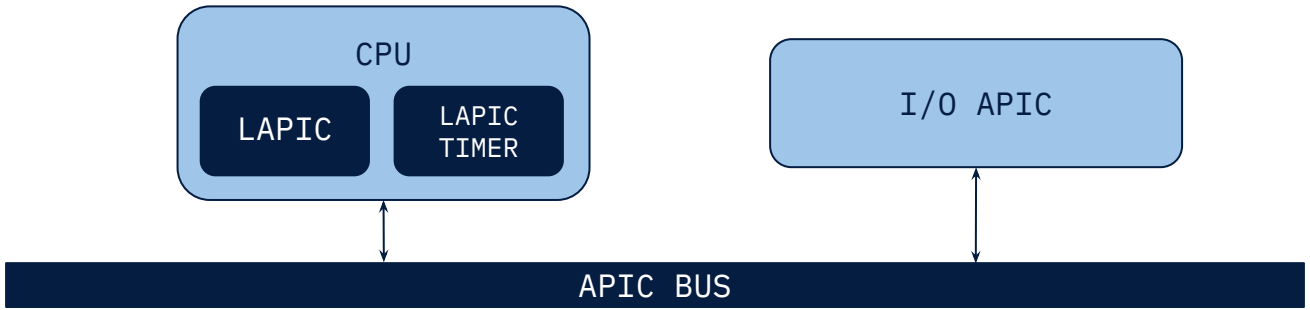
- This complexity makes them difficult to extend
  - Multicore support was added retrospectively to most kernels, using big kernel locking (BKL) as a stepping stone
  - New scheduling policies can utilise system resources more efficiently, but rarely make their way into production operating systems

Key contributions

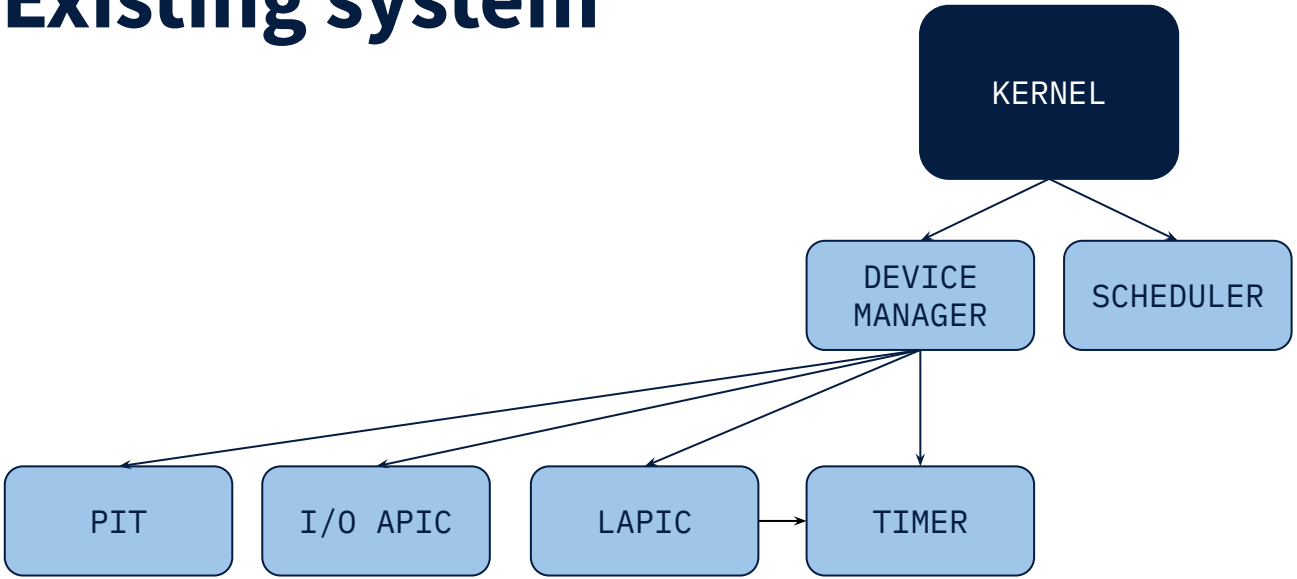
- A **teaching** tool
  - For understanding multicore scheduling and multithreaded programming
- A **research** tool
  - For investigating new, dynamic scheduling approaches
- A **technical** tool
  - That can be downloaded and booted on real hardware by anyone

Existing system

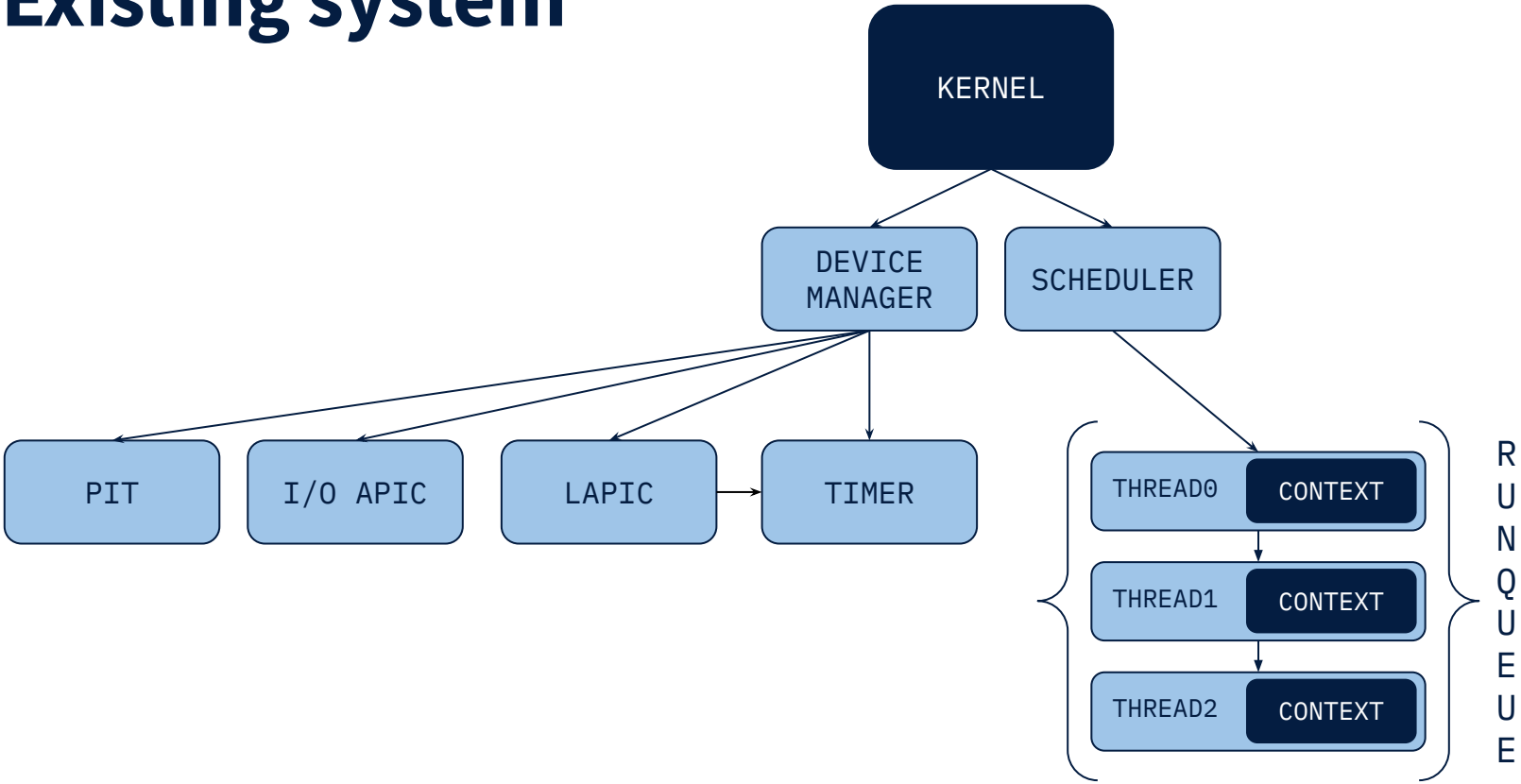
- InfOS follows the APIC standard
- Two interrupt controllers
  - I/O APIC, directs external interrupts, e.g. keyboard strokes
  - LAPIC, responsible for internal interrupts, e.g. timer



Existing system

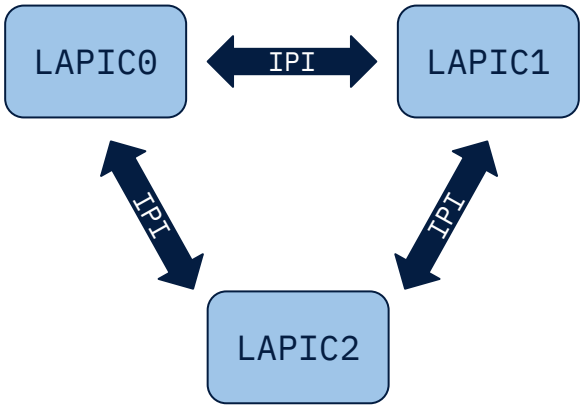


Existing system



Intel initialisation protocol

- Two classes of processors
  - Bootstrap processor (BSP)
  - Application processors (APs)
- Communicate via interrupts
  - LAPICs send interprocessor interrupts (IPIs) to one another



Detecting cores

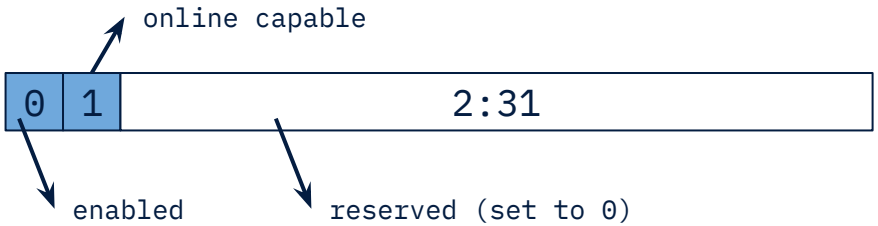
- The MADT table describes all interrupt controllers in the system
  - Entry type 0 represents LAPICs

Offset (hex)	Length	Description
2	1	ACPI Processor ID
3	1	APIC ID
4	4	Flags

Detecting cores

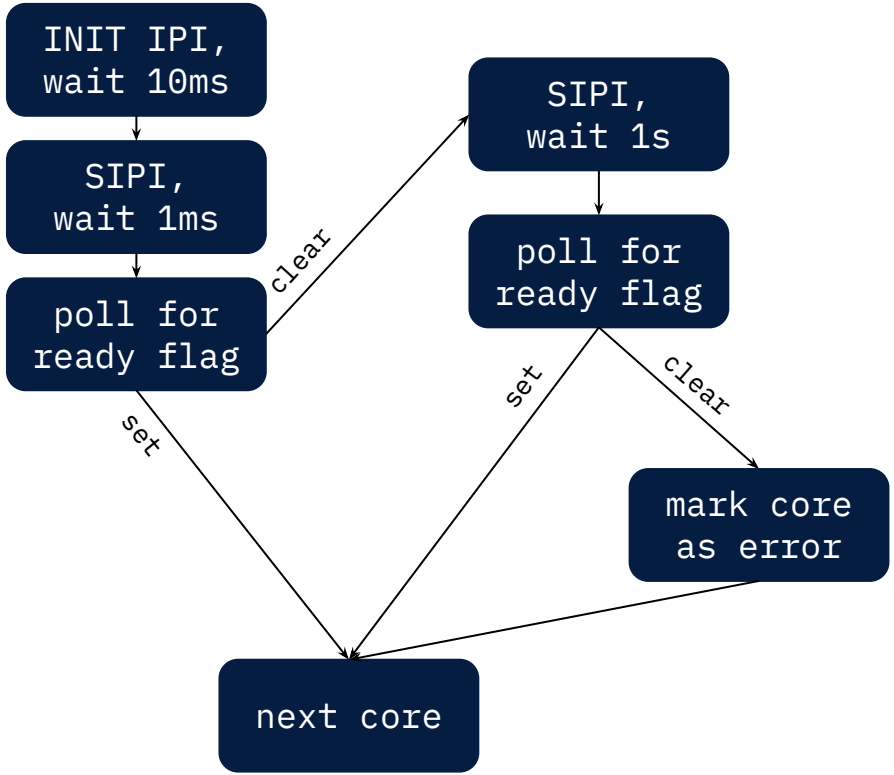
- The MADT table describes all interrupt controllers in the system
  - Entry type 0 represents LAPICs

Offset (hex)	Length	Description
2	1	ACPI Processor ID
3	1	APIC ID
4	4	Flags

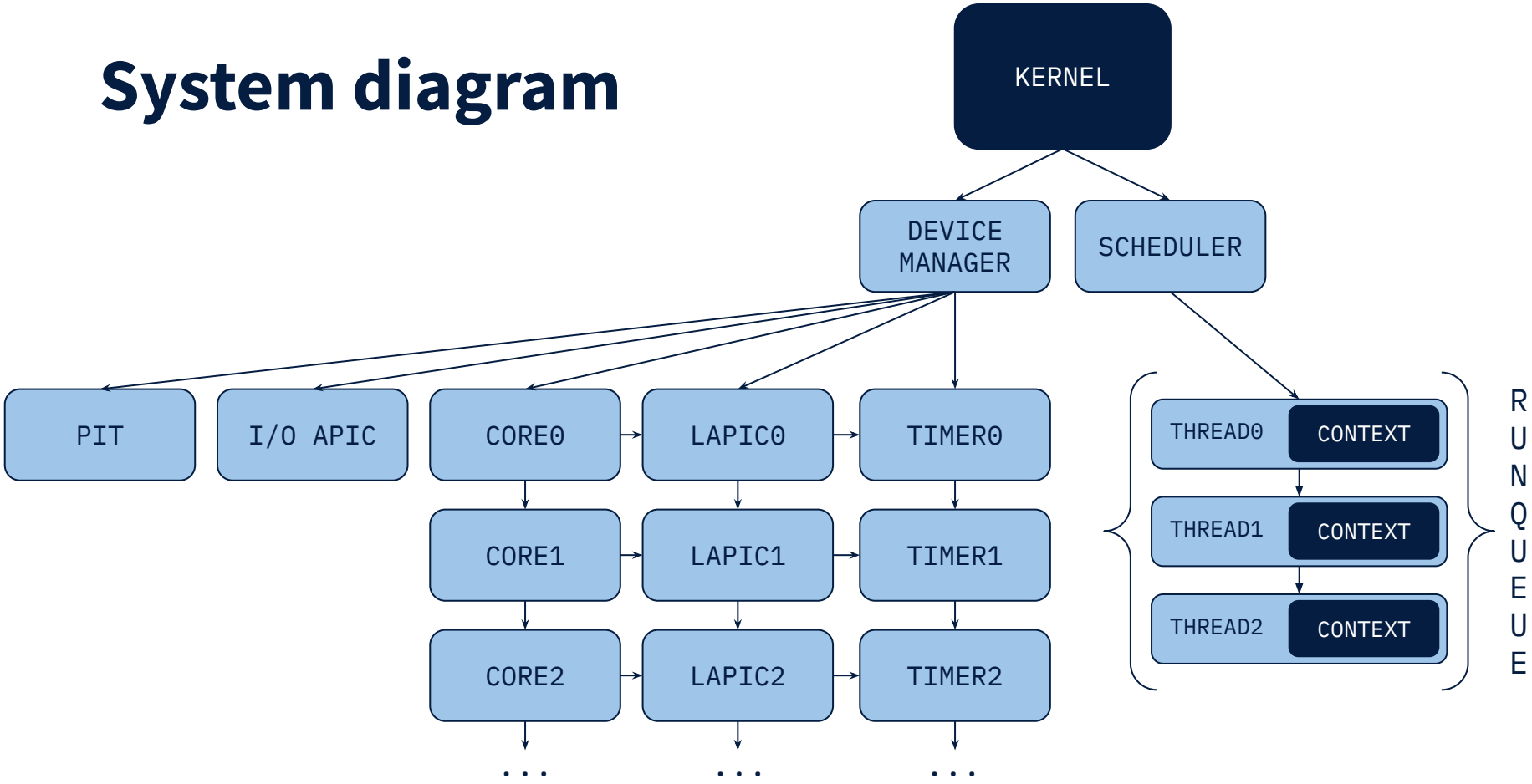


Initialising cores

- To start **each** AP, the BSP sends a sequence of IPIs with delays
- INIT-SIPI-SIPI sequence

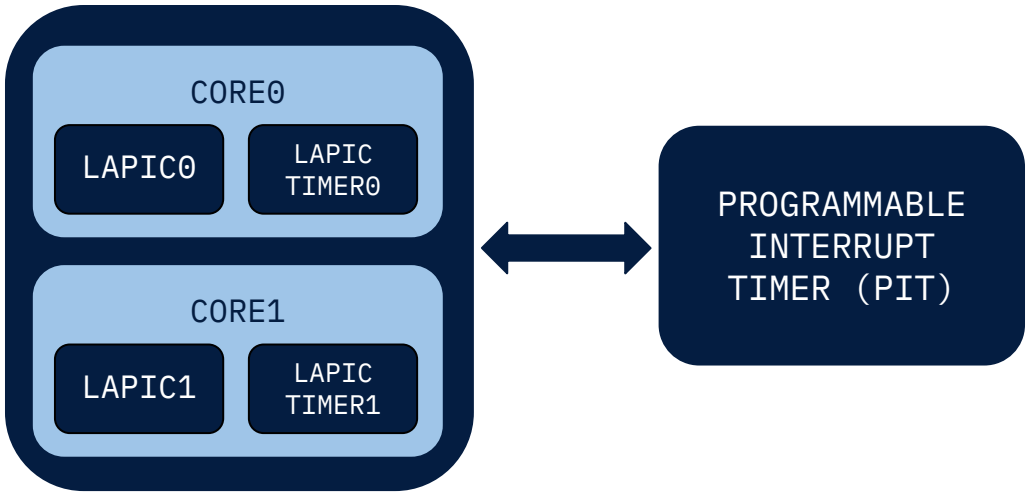


System diagram

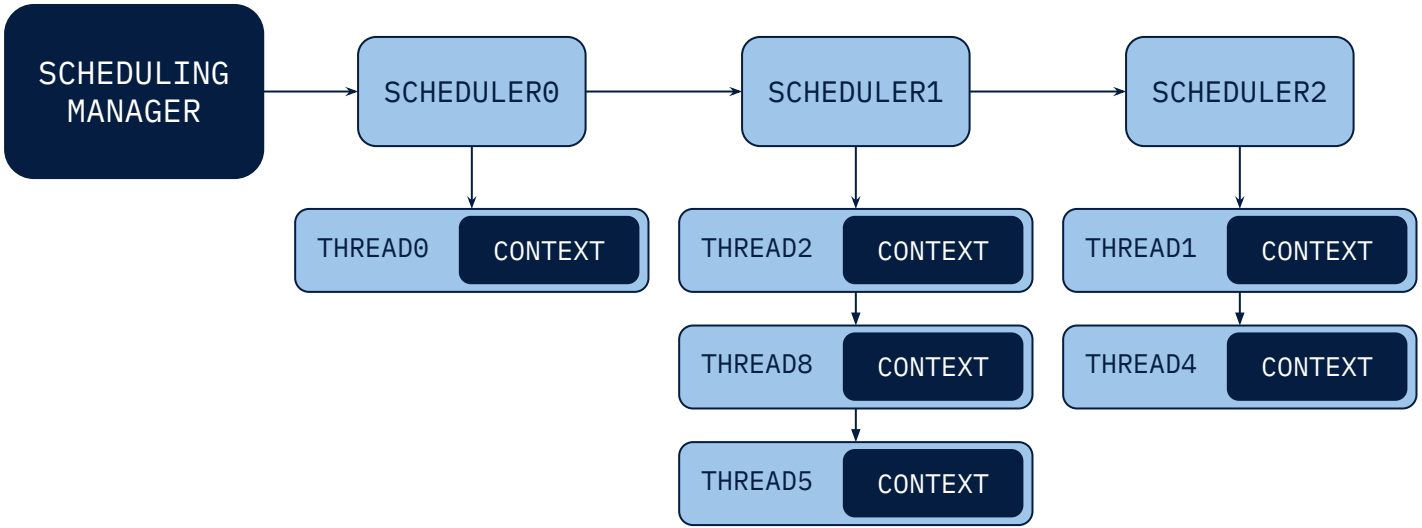


Calibrating timers

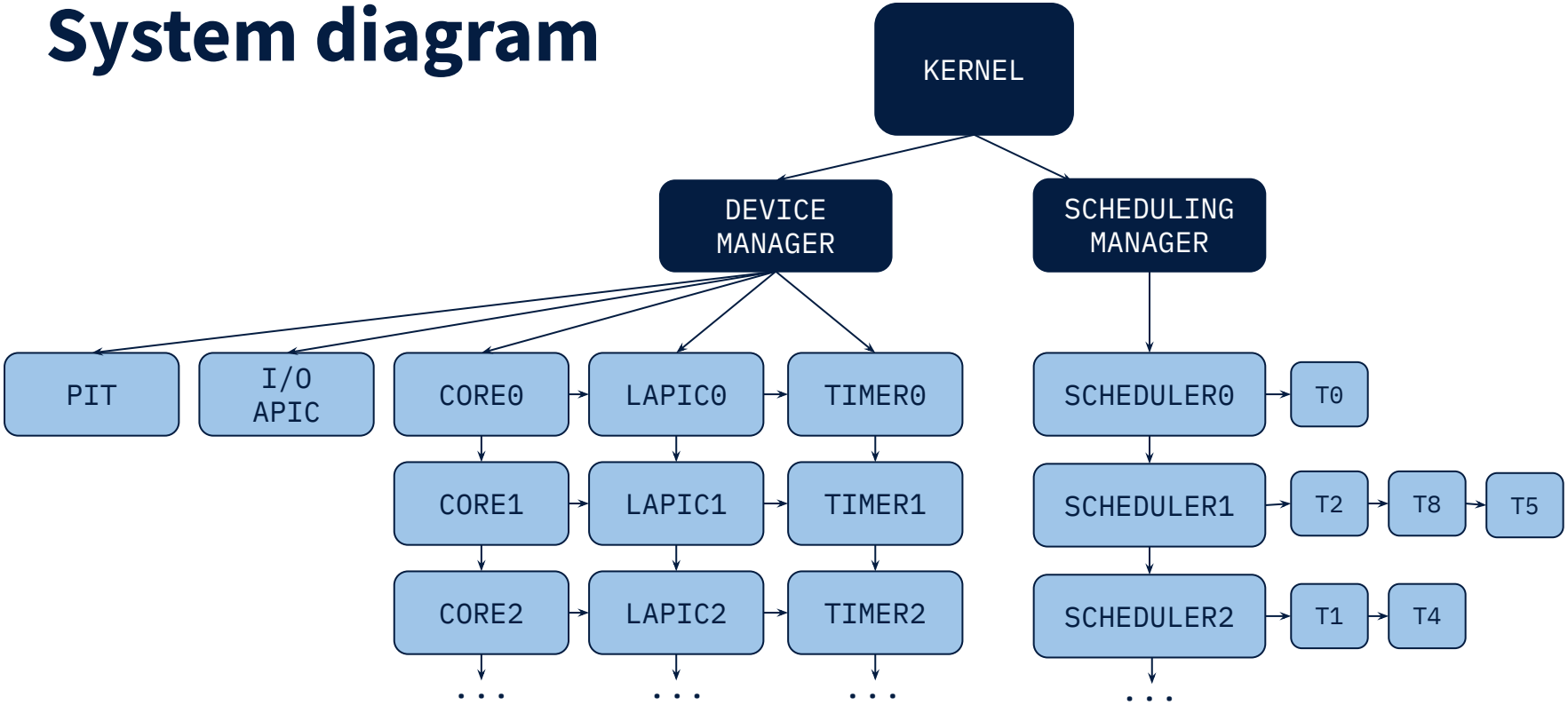
- InfOS supports preemptive scheduling using timer interrupts
  - The frequency of local timers can vary between machines
  - Local timers need to be calibrated before scheduling begins
  - This can be done with a **known reference** timer, e.g. PIT



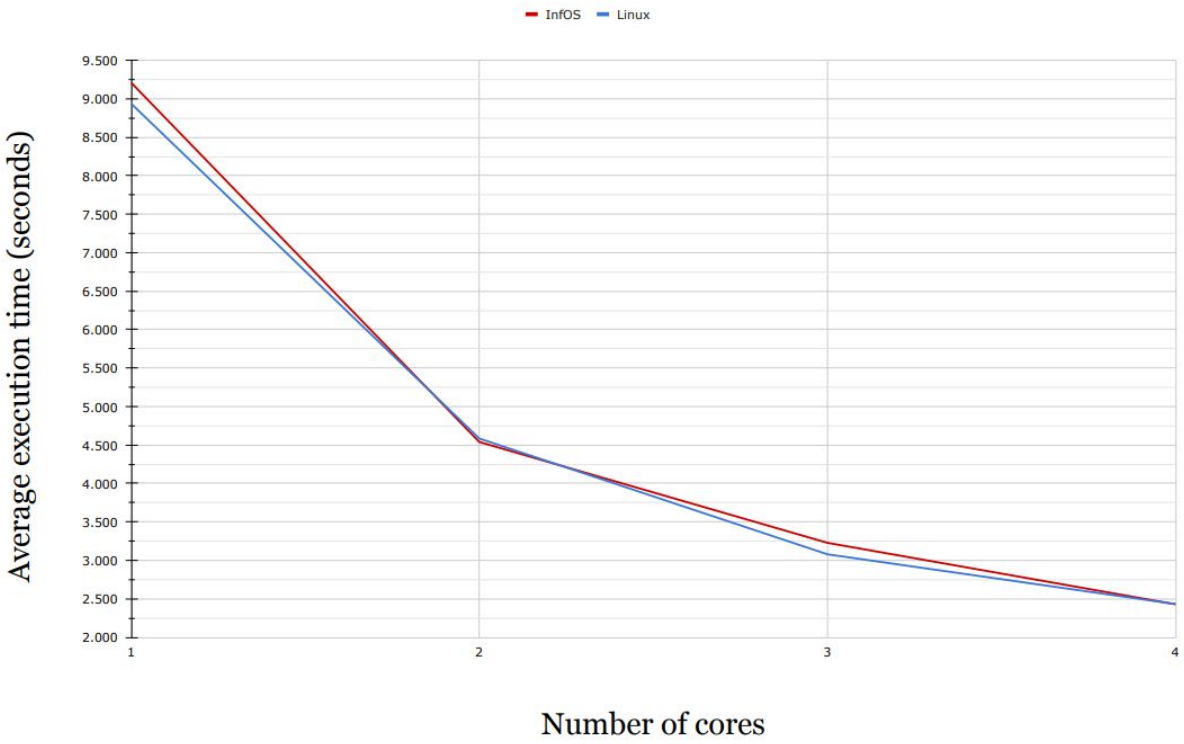
Scheduling threads



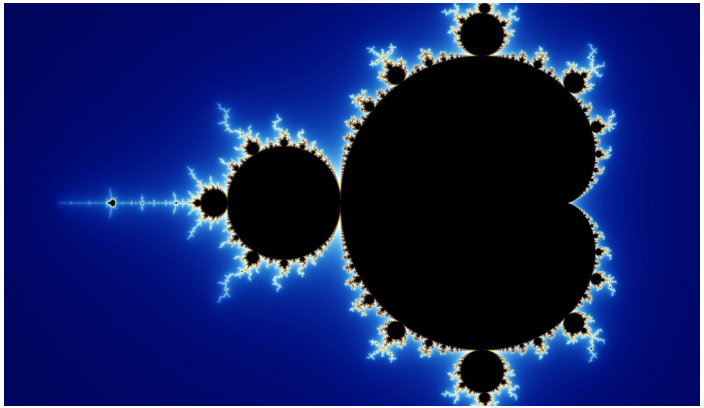
System diagram



Evaluation

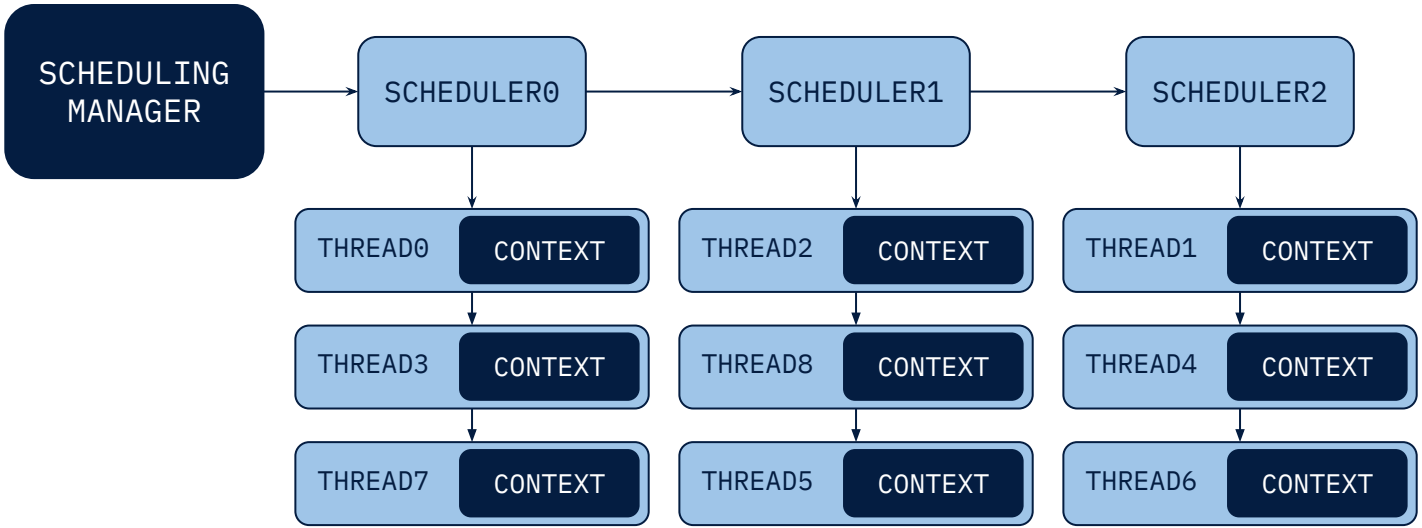

InfOS
Linux
9.500
9.000
8.500
8.000
7.500
7.000
6.500
6.000
5.500
5.000
4.500
4.000
3.500
3.000
2.500
2.000
1
2
3
4
Average execution time (seconds)
Number of cores


TightVNC: QEMU



Future work

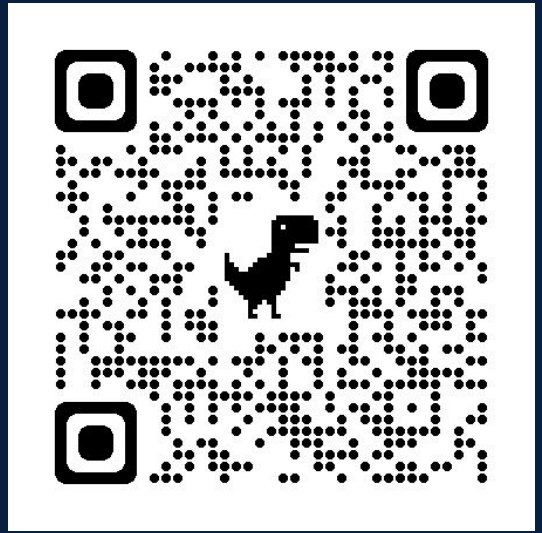
- Experiment with new scheduling approaches
- Teach undergraduates about multicore scheduling!



Thank you!

Any questions?

kimbethstonehouse/multicore-support



Kim.Stonehouse@ed.ac.uk

THE UNIVERSITY of EDINBURGH