OS Overview / Process and Thread

국민대학교 임베디드 연구실
경 주 현

오늘 강의 계획

- **OS 기본 이론 및 강의 내용 설명**

- **리눅스 기본 이론**
  - 성능: 멀티코어, 리눅스 Lock and scalability
  - 파워: 리눅스 PM, Timer 기반 DVFS

- **리눅스 스케줄러**
  - 리눅스 스케줄러 class, 트레이스(실습)
  - CFS, CFS Parameters, Load Balancer
  - Group Scheduling, Bandwidth Control, PELT
  - EAS(Energy Aware Scheduling)
  - Android PM, Deadline scheduler

Goals

Understand Linux Scheduler/DVFS in detail by designing and implementing

What do apps want from a O/S?

What do applications want from a O/S?

- Abstract the hardware for convenience and portability
- Multiplex the hardware among multiple applications
- Isolate applications to contain bugs
- Allow sharing among applications

What services does a O/S typically provide?

What services does a O/S typically provide?

- Processes
- Memory
- File contents
- Directories and file names
- Security
- Many others: users, IPC, network, time, terminals

What is key requirements for kernels?

Key requirements for kernels

- Isolation
- Multiplexing
- Interaction

Focus on isolation

Constraining requirement.

What is isolation?

- **Enforced separation to contain effects of failures**

- **The process is the usual unit of isolation**

What is isolation?

- **Prevent process X from wrecking or spying on process Y**
- **Prevent a process from wrecking the operating system itself**
- **Prevent a malice as well as bugs**

In order to isolate,

In order to isolate

- Apps must use OS interface
- Apps cannot directly interact with hardware
- Apps cannot harm operating system
- Apps cannot directly affect each other

Processor

Processors provide mechanisms

- Hardware provides user mode and kernel mode
- Some instructions can only be executed in kernel mode
- Device access, processor configuration, isolation mechanisms

Hardware isolation mechanisms

- **Operating systems runs in kernel mode**
  - kernel is a big program
  - services: processes, file system, net
  - low-level: devices, virtual memory
  - all of kernel runs with full hardware privilege
    - convenient

- **Applications run in user mode**
  - Kernel sets up per-process isolated address space
  - **System calls** switch between user and kernel mode

O/S

How to isolate process memory?

How to isolate process memory?

- **Address Space**
  - Give memory - code, variables, heap, stack
- **prevent it from accessing other memory**
  - kernel or other processes
- **"paging hardware" in the memory management unit (MMU)**
- **MMU translates (or "maps") every address issued by program**
- **VA -> PA**

How to isolate CPU?

How to isolate CPU?

- **H/W provides a periodic "clock interrupt"**
  - Forcefully suspends current process
  - Jumps into kernel
  - Kernel can switch to a different process
- **Preventing long computations, or buggy infinite loops**
- **Kernel must save/restore process state (registers)**
  - Context switch

Process

- An abstract virtual machine
- As if it had its own CPU and memory
- Not accidentally affected by other processes.
- Motivated by isolation

Thread

- An abstraction that contains enough state of a running program
- It can be stopped and resumed
- Multiple threads share an address space

Overview of switch between two threads

User -> kernel transition (system call or timer)

Kernel -> kernel switch

Kernel -> user transition

Conclusion

Apps requirements -> O/S -> Isolation -> scheduler -> process/thread

Reference

- https://pdos.csail.mit.edu/6.828/2016/schedule.html
- http://web.mit.edu/6.033
- http://www.rdrop.com/~paulmck/
- "**Is Parallel Programming Hard, And If So, What Can You Do About It?"**
- Davidlohr Bueso. 2014. Scalability techniques for practical synchronization primitives. *Commun. ACM* 58

http://queue.acm.org/detail.cfm?id=2698990

- **"CPUFreq and The Scheduler Revolution in CPU Power Management", Rafael J. Wysocki**
- **https://sites.google.com/site/embedwiki/oses/linux/pm/pm-qos**
- **https://intl.aliyun.com/forum/read-916**
- **User-level threads : co-routines**

  http://www.gamedevforever.com/291

  https://www.youtube.com/watch?v=YYtzQ355_Co

- **Scheduler Activations**
  - https://cgi.cse.unsw.edu.au/~cs3231/12s1/lectures/SchedulerActivations.pdf
- **https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics)**
- **http://jake.dothome.co.kr/**
- **http://www.linuxjournal.com/magazine/completely-fair-scheduler?page=0,0**
- **https://www2.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/6_CPU_Scheduling.html**
- **"Energy Aware Scheduling", Byungchul Park, LG Electronic**
- **"Update on big.LITTLE scheduling experiments", ARM**
- **"EAS Update" 2015 september ARM**
- **"EAS Overview and Integration Guide", ARM TR**
- **"Drowsy Power Management", Matthew Lentz, SOSP 2015**
- **https://www.slideshare.net/nanik/learning-aosp-android-hardware-abstraction-layer-hal**
- https://www.youtube.com/watch?v=oTGQXqD3CNI
- https://www.youtube.com/watch?v=P80NcKUKpuo
- https://lwn.net/Articles/398470/
- "SCHED_DEADLINE: It's Alive!", ARM, 2017