ECE 692
Power-Aware Computer Systems

Process Scheduling in OS and the Power Problem

Prof. Xiaorui Wang

Grading
- Homework (Critique) 20%
- Class presentation: 15%
- Final exam: 10%
- Group project 50%
  - Proposal presentation 5%
  - Midterm presentation 10%
  - Final presentation 15%
  - Final report 20%
- Participation: 5%
  - Attend the classes
  - Contribute to class discussion
  - Feel free to ask questions (during project/class presentations)

Outline
- Operating systems
- Processes
- Context switch
- Process scheduling
- The power problem

Basic Functions of OS
- OS controls resources:
  - who gets the CPU;
  - when I/O takes place;
  - how much memory is allocated.
- Application programs run on top of OS services
- Challenge: manage multiple, concurrent tasks.

Process
- A process is a unique execution of a program.
  - Several copies of a program may run simultaneously or at different times.
- A process has its own context:
  - Data in registers, PC, status, memory.
  - Stored in activation record
- OS manages processes.

Processes and CPUs
- Activation record
  - process context.
- Context switch:
  - current CPU context goes out;
  - new CPU context goes in.
Co-Operative Multitasking

- Process management:
  - hides context switching mechanism;
  - relies on processes to voluntarily give up CPU.
- Each process allows a context switch at cswitch() call.
- Separate scheduler chooses which process runs next.

```
if (x > 2)
  sub1(y);
else
  sub2(y, z);
cswitch();
proc_data(r, s, t);
cswitch();
```

Problems with Co-Operative Multitasking

- Programming errors can keep other processes out:
  - process never gives up CPU;
  - process waits too long to switch, missing input.

```
Process2() {
  x = global1; /* global1 is an input to the process */
  while (x < 500)
    x = aproc(global2); /* subroutine does its work */
cswitch();
}
```

Preemptive Multitasking

- No more voluntary release of CPU
  - Operating System (OS) is now in charge
- Most powerful form of multitasking:
  - OS controls when context switches;
  - OS determines what process runs next.
- Use periodic timer interrupts to call OS to switch contexts

```
interrupt
CPU
P1    OS
interrupt
P1    OS
P2
```

Preemptive Context Switching

- Timer interrupt gives control to OS, which saves interrupted process’s state in an activation record.
- OS chooses next process to run.
- OS installs desired activation record as current CPU state.

Process Scheduling

- Who gets CPU?
  - Usually priority-based
    - Priority can be based on deadline, requested time, or...
- When context switch occurs?
  - Low priority tasks are running while high priority tasks are waiting
- We have three classes to introduce different scheduling algorithms
  - Key issue of real-time systems

Process States

- A process can be in one of three states:
  - executing on the CPU;
  - ready to run;
  - waiting for data.
Process Management
- OS keeps track of:
  - process priorities;
  - scheduling state;
  - process control block.
- Processes may be created:
  - statically before system starts;
  - dynamically during execution.
- OS controls when context switches and what process runs.

Process Scheduling
Embedded vs. General-Purpose
- Workstations try to avoid starving processes of CPU access.
  - Fairness = access to CPU.
- Embedded systems must meet deadlines.
  - Low-priority processes may not run for a long time.

Priority-Driven Scheduling
- Every process has a priority.
- CPU goes to the highest-priority ready process
- Variants
  - Fixed vs. dynamic priority
  - Preemptive vs. non-preemptive

Scheduling Example
- Each process has a fixed priority (1 highest):
  - P1: priority 1; P2: priority 2; P3: priority 3.
  - P2 released
  - P1 released
  - P3 released

Preemptive Priority Scheduling
- Most common real-time scheduling approach
  - Real-Time POSIX
  - Real-time priorities in Linux, Solaris, and Windows
  - Most RTOS: VxWorks ...
- Not the only possible way
  - Clock-driven scheduling (e.g., round robin)
  - Reservation-based scheduling
  - Proportional share scheduling
  - FIFO scheduling

The Scheduling Problem
- Can we meet all deadlines?
  - Must be able to meet deadlines in all cases.
- How much CPU horsepower do we need to meet our deadlines?
- Timing violations: What happens if a process doesn’t finish by its deadline?
  - Hard deadline: system fails if missed.
  - Soft deadline: user may notice, but system doesn’t necessarily fail.
Outline
• Operating systems
• Processes
• Context switch
• Process scheduling
• The power problem

The Power Problem
• Microprocessors improve performance at the cost of power!
  • Performance/watt remains low.
• Solutions
  • Microprocessors offer features (hardware support) for controlling power consumption.
  • Software performs power management.

Power vs. Energy
• Power: Energy consumed per unit time
  • 1 watt = 1 joule/second
  • Heat depends on power consumption.
  • Battery life depends on energy consumption.

Outline
• Hardware support
• Power management policy
• Power manager
• Holistic approach

CMOS Power Consumption
• All digital systems are built with CMOS
• Voltage drops
  • Power consumption of a CMOS is proportional to the square of the power supply voltage ($V^2$).
• Toggling
  • CMOS uses higher power when having more activity
• Leakage
  • Even when CMOS is inactive, some charge leaks out of the circuit’s nodes

General Power-Saving Features
To deal with voltage drops
  • Reduce power supply voltage to the lowest level that provides required performance
To deal with toggling
  • Run at lower clock frequency (slower)
  • Reduce activity by disabling function units when not in use.
To deal with leakage
  • Eliminate leakage current by disconnecting parts from power supply when not in use.
Clock Gating
- Applicable to clocked digital components
  - Processors, controllers, memories
- Stop the clock → stop signal propagation in circuits
- Pros
  - Simple
  - Very short transition time if
    - Only clock distribution is stopped while clock generation is not stopped
- Cons
  - Clock itself still consumes energy
  - Cannot prevent power leaking – exist as long as power supply is connected

Supply Shutdown
- Disconnect parts from power supply when not in use.
- Pros
  - General
  - Save most power
- Cons
  - Very long transition time

Dynamic Voltage Scaling
- Why voltage scaling?
  - Power ∝ V^2
  - Reduce power supply voltage → save energy
  - Lower clock frequency allows lower voltage
  - Tradeoff between performance and battery lifetime
- Why dynamic?
  - Power consumption is not a constant.
    - Depending on current workloads.
  - Peak computing rate is usually much higher than average.
  - Have to be dynamic to respond to power consumption variations.

Dynamic Voltage Scaling (cont.)
- Changing voltage takes time
  - Need to stabilize power supply and clock
- Continuous and discrete voltage are both possible
  - Many microprocessors have discrete power modes

Outline
- Hardware support
- Power management policy
- Power manager
- Holistic approach

Power Management Styles
- Static Power Management
  - Does not depend on activity.
  - Example: user-activated power-down mode.
    - Laptop monitor changes to low-power state when unplugged
- Dynamic Power Management (DPM)
  - Automatic action based on activity.
  - Example: automatically disabling function units.
    - Monitor automatically turns off after no activity for a certain period of time
**Dynamic Power Management**

- **Goals:** Energy conservation AND good performance
- **Need tradeoff between conflicting goals!**
- **Fundamental premises**
  - Systems have varying workloads during operation
    - Running workloads: higher power and better performance
    - Idling: lower power and worse performance
  - It is possible to predict the fluctuations of workload with some degree of accuracy
    - Daytime vs. nighttime
- **Performed by a Power Manager**
  - React to or predict workload variations

---

**Problem Formulations of Dynamic Power Management**

- Minimize power under performance constraints
  - Real-time constraint
- OR
  - Optimize performance under power constraints
    - Battery lifetime constraint
    - Best server performance under the capacity of the power supply

---

**Cost of Dynamic Power Management**

- Power management is not free
- Going into/out of an inactive mode costs:
  - time;
  - energy.
- Must determine if going into mode is worthwhile.
- Can model CPU power states with Power State Machine (PSM)

---

**PSM Example: SA-1100**

- SA-1100 is a StrongARM processor from Intel
  - Designed to provide sophisticated power management capabilities controlled by the on-chip power manager
- Three power modes:
  - Run: normal operation.
  - Idle: stops CPU clock, with I/O logic still powered.
  - Sleep: shuts off most of chip activity

---

**SA-1100 Sleep**

- **RUN → SLEEP**
  - (30 μs) Flush to memory CPU states (registers)
  - (30 μs) Reset processor state and wakeup event
  - (30 μs) Shut down clock
- **SLEEP → RUN**
  - (10 ms) Ramp up power supply
  - (150 ms) Stabilize clock
  - (negligible) CPU boot
- Overhead of sleep to run is much larger

---

**SA-1100 Power State Machine**

```
run
  10 μs
  160 ms

idle
  10 μs

sleep
  90 μs
```

- \( P_{\text{run}} = 400 \text{ mW} \)
- \( P_{\text{idle}} = 50 \text{ mW} \)
- \( P_{\text{sleep}} = 0.16 \text{ mW} \)

\[ \text{Power consumption during transition} = P_{\text{run}} \]

---
Summary

- Operating systems
- Process states and management
- Context switch
- Process scheduling
- The power problem