Energy Management
Dynamic Voltage & Frequency Scaling (DVFS)

Advanced Operating Systems
Reza Azimi
Shiraz University, Spring 2010

Background

- Dynamic Power
  - a.k.a “switching power”
  - Power dissipation due to capacitance charging at transitions from 0->1 and 1->0
  - Can be reduced by decreasing switching frequency

- Static Power
  - a.k.a “leakage power”
  - steady, per-cycle energy cost
  - Can be reduced only by turning off a component

- Dynamic power dominates, but static power has gained importance as clock frequency has increased.
Dynamic CMOS Power Dissipation

- Capacitance: Function of wire length, transistor size
- Supply Voltage: Has been dropping with successive fab generations
- Power \( \sim \frac{1}{2} CV^2 Af \)
- Activity factor: How often, on average, do wires switch?
- Clock frequency: Increasing...

Past, Present, Future

- Past:
  - Power important only for laptops, PDAs, sensors, cell phones

- Present:
  - Power a Critical, Universal design constraint even for very high-end chips

- Future:
  - Circuits and process scaling can no longer solve all power problems.
  - SYSTEMS must also be power-aware
    - Architecture, OS, compilers

ref: Martonosi, Brooks, Bose Tutorial at HPCA 2001
Energy vs. Power

- **Energy (J)**
  - Focusing on the amount of energy required workload
  - Common metric: energy-per-instruction
  - Used for analyzing energy efficiency

- **Power (W)**
  - Focusing on maximum (or average) power a system requires to operation
  - Used for circuit/system design or for electric billing

- **Joining Performance & Power**
  - energy * delay  (EDP)
  - energy * (delay)^2  (ED^2 P)
  - Operations per second per watt

System Power-Management

- Micro-Architecture & Architecture
  - Shrink structures
  - Shorten wires
  - Reduce activity factors
  - Improve instruction-level control

- Compilers
  - Reduce wasted work: "standard" operations
  - More aggressive register allocation and cache optimization
  - Trade off parallelism against clock frequency

- Operating Systems
  - Natural, since OS is traditional resource manager
  - Equal energy scheduling
  - Battery-aware or thermally-aware adaptation

---

*ref: Martonosi, Brooks, Bose Tutorial at HPCA 2001*
Reducing Dynamic Power

- Reduce capacitance
  - Simpler, smaller design

- Reduce activity
  - Smarter design
  - Reduced IPC

- Reduce frequency
  - Often in conjunction with reduced voltage

- Reduce voltage
  - Biggest impact due to quadratic effect, widely employed
  - Mostly done dynamically

Dynamic Voltage Scaling (DVS)

- The ability to change the CPU clock frequency and/or its operational voltage at runtime.
  - Also referred to as Dynamic Voltage & Frequency Scaling (DVFS)

- Supported by most mainstream processors
  - AMD PowerNow! technology
  - Intel SpeedStep technology
OS-Level Energy Management

- Dynamic Clock Scheduling
  - DVS

- Reducing Paging Activities
  - e.g., Power-aware Virtual Memory

- Reducing I/O Power Consumption
  - e.g., Cooperative I/O

- etc.

Scheduling for Reduced CPU Energy

- Work by M. Weiser et. al at Xerox
- Targets mainly PDAs and pocket computers

- General Approach
  - Goal
    - Fill idle periods with running at slower speed
  - Energy Saving Technique
    - The fine grain control of CPU clock speed
    - Running slower and at reduced voltage
  - Evaluation
    - Using trace-driven simulation
    - Workloads: software developments, documentation, emails, typing, scrolling, etc.
Weiser’s Early Work Assumptions

- Trace-based Simulation
- No reordering of trace data events
- Using no energy when idle
  - Only taking dynamic power into account
  - Is this optimistic or pessimistic?
- Taking no time to switch speeds
  - Generally not true, but going to that direction

Scheduling Algorithms

- **OPT (unbounded-delay perfect-future)**
  - Taking the entire trace
  - Stretching all the run times to fill all the idle times
  - Imaginary batch job with perfect knowledge
  - Impractical & undesirable
  - Bad response time

- **FUTURE (bounded-delay limited-future)**
  - Taking the future trace of a small window
  - Window sizes: 1 ms ~ 400 sec
  - Impractical but desirable
  - Good response time on a window of 10 to 50 ms
Scheduling Algorithms (cont’d)

- **PAST (bounded-delay limited-past)**
  - Looking a fixed window into the past, assuming the next window will be like the previous one
  - Examine % busy during the previous interval and adjust speed for the next interval
  - Excess cycles can build up if speed (+voltage) is set too low. => Penalty metric

- **Dealing with Excess Cycles**
  - At each interval, count up left over cycles that accumulated because you ran too slow
  - Switch to full speed if there were more excess cycles than idle time in the previous interval
  - Hard idle (page fault, disk request) cannot be squeezed

---

Scheduling Algorithms (cont’d)

- Presented by K. Govil et. al, 1995

- Looking back to N last intervals (as opposed to 1)
  - Calculating exponentially weighted AVG over a moving window of N intervals

\[
W_t = \frac{(N \times W_{t-1} + U_{t-1})}{(N+1)}
\]
Trace Driven Simulation

- **Trace Points**
  - Sched: context switch away a process
  - Idle on: enter the idle loop
  - Idle off: leave idle loop to run a process
  - Fork: create a new process
  - Exec: overlay a new process with another program
  - Exit: process termination
  - Sleep: wait on an event
  - Wakeup: notify a sleeping process

- **Traces**
  - Short runs during specific tasks, editing etc.
  - Long runs of several hours

Interval Scheduling

- Algorithms (when):
  - Past
  - AVG
- Stepping (how much):
  - One
  - Double
  - Peg – min or max
- Based on unfinished work during previous interval

---

*Advanced Operating Systems, Shiraz University, Winter 88/Spring 89, Reza Azimi*
### Results & Discussion

- **Potential energy saving (up to 70%)**
  - 10-50ms intervals

### Discussion

- OPT and FUTURE are impractical to implement
- PAST is not practical to implement without knowledge about application behavior
- Too sensitive to the interval size
  - Varies for different applications
- Too low a minimum speed can be harmful
  - Resulting to have lots of excessive cycles

---

### Policies for Dynamic Clock Scheduling

- Presented by Grunwald et al, OSDI 2000

- Experimental study (not simulation) – based on ITSY hardware with Linux
  - StrongARM 59MHz to 206MHz, 1.5v or 1.23V

- Measured whole system power consumption
  - Important to capture interactions with other power consuming components of hardware platform

- Inelastic performance constraints – don’t want to allow user to see any performance degradation

- Realistic workload (for handheld device): MPEG player, Web viewing, Chess game, Talking editor
Implementation

- Capturing utilization measure
  - Start with no a priori information about applications and need to dynamically infer/predict behavior (patterns/“deadlines”/constraints?)
  - Idle process or “real” process – usually each quantum is either 100% idle or busy
  - $\text{AVG}_N$: weighted utilization at time $t$
    $$W_t = \frac{NW_{t-1} + U_{t-1}}{N+1}$$

- Adjusting the clock speed
  - Idea is to set the clock speed sufficiently high to meet deadlines (but deadlines are not explicit in algorithm)

Results and Conclusions

- It is hard to find any discernible patterns in “real” applications
  - Better at larger time scales (corresponding to larger windows in $\text{AVG}_N$) but then systems becomes unresponsive
  - Poor coupling between adaptive decisions of applications themselves and system decision-making (example: MPEG player that either blocked or spun)
  - NEED application-supplied information

- Simple averaging shows asymmetric behavior – clock rate drops faster than ramps up

- $\text{AVG}_N$ could not stabilize on the “right” clock speed - Occillations