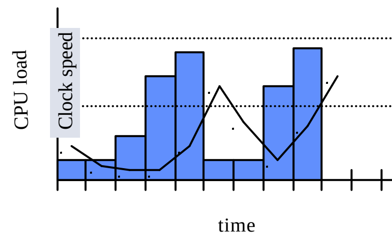
Dynamic Voltage Scaling

The question: at what clock rate/voltage should the CPU run in the next scheduling interval?

- Voltage scalable processors
  - StrongARM SA-2 (500mW at 600MHz; 40mW at 150MHz)
  - Speedstep Pentium III
  - AMD Mobile K6 Plus
  - Transmeta
- Power is proportional to $V^2 \times F$
- Energy will be affected
  (+) by lower power,
  (-) by increased time

Interval Scheduling
(adjust clock based on past window, no process reordering involved)



Weiser et. al.

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

Policies for Dynamic Clock Scheduling
Grunwald et al, OSDI 2000

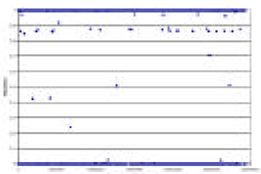
- Experimental study (not simulation) – based on ITSY hardware with Linux
  - StrongARM 59MHz to 206MHz, 1.5v or 1.23V
- Measured whole system power consumption
  - Important to capture interactions with other power consuming components of hardware platform
- Inelastic performance constraints – don't want to allow user to see any performance degradation
- Realistic workload (for handheld device): MPEG player, Web viewing, Chess game, Talking editor

Implementation of Scheduling Algorithms

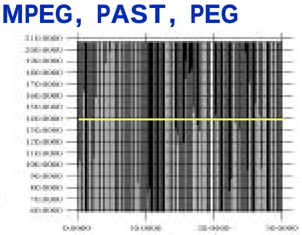
Issues:

- 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
  - AVG_N: weighted utilization at time t
    $W_t = (NW_{t-1} + U_{t-1}) / (N+1)$
- Adjusting the clock speed
  - Idea is to set the clock speed sufficiently high to meet deadlines (but deadlines are not explicit in algorithm)

Best Result – MPEG, PAST, PEG



(a) MPEG Program at 206MHz



Algorithm	Energy
Constant Speed @ 206.4 MHz, 1.5 Volts	85.59 - 86.40
Constant Speed @ 132.7 MHz, 1.5 Volts	79.59 - 80.94
Constant Speed @ 132.7 MHz, 1.23 Volts	73.76 - 74.41
PAST, Peg - Peg, Thresholds: > 98% scales up, < 93% scales down, 1.5 Volts	85.03 - 85.47
PAST, Peg - Peg, Thresholds: > 98% scales up, < 93% scales down, Voltage Scaling @ 162.2 MHz	84.60 - 85.45

Results and Conclusions

- It is hard to find any discernible patterns in "real" applications
  - Better at larger time scales (corresponding to larger windows in 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
- AVG_N could not stabilize on the "right" clock speed - Occillations

Negative Results are a Hard-Sell

Discussion: Is this study and its results *satisfying*?

Voltage Scheduling in the lpARM Microprocessor System
Pering et. al. ISLPED 2000

Read for next time

Background: Liu and Layland (classic TR Scheduling paper)

Hard real time - tasks executed in response to events (requests) and must be completed in some fixed time (deadline)

Soft real time - statistical distribution of response times

Assumptions

Tasks are periodic with constant interval between requests, T_i (request rate $1/T_i$)

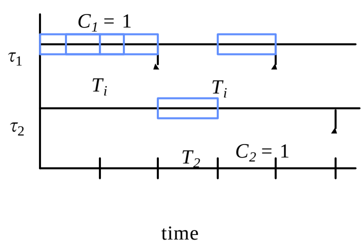
Each task must be completed before the next request for it occurs

Tasks are independent

Run-time for each task is constant (max), C_i

Any non-periodic tasks are special

Task Model



Definitions

Deadline is time of next request

Overflow at time t if t is deadline of unfulfilled request

Feasible schedule - for a given set of tasks, a scheduling algorithm produces a schedule so no overflow ever occurs.

Critical instant for a task - time at which a request will have largest response time.

- Occurs when task is requested simultaneously with all tasks of higher priority

Rate Monotonic

- Assign priorities to tasks according to their request rates, independent of run times
- Optimal in the sense that no other fixed priority assignment rule can schedule a task set which can not be scheduled by rate monotonic.
- If feasible (fixed) priority assignment exists for some task set, rate monotonic is feasible for that task set.

Earliest Deadline First

- Dynamic algorithm
- Priorities are assigned to tasks according to the deadlines of their current request
- With EDF there is no idle time prior to an overflow
- For a given set of *m* tasks, EDF is feasible iff

  $C_1/T_1 + C_2/T_2 + \ldots + C_m/T_m \leq 1$
- If a set of tasks can be scheduled by any algorithm, it can be scheduled by EDF