Dynamic Voltage Scaling

CPU can run at different clock frequencies/voltage:

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

Dynamic Voltage Scheduling

Questions addressed by the scheduler:

- Which process to run
- When to run it
- How long to run it for
- How fast to run the CPU while it runs

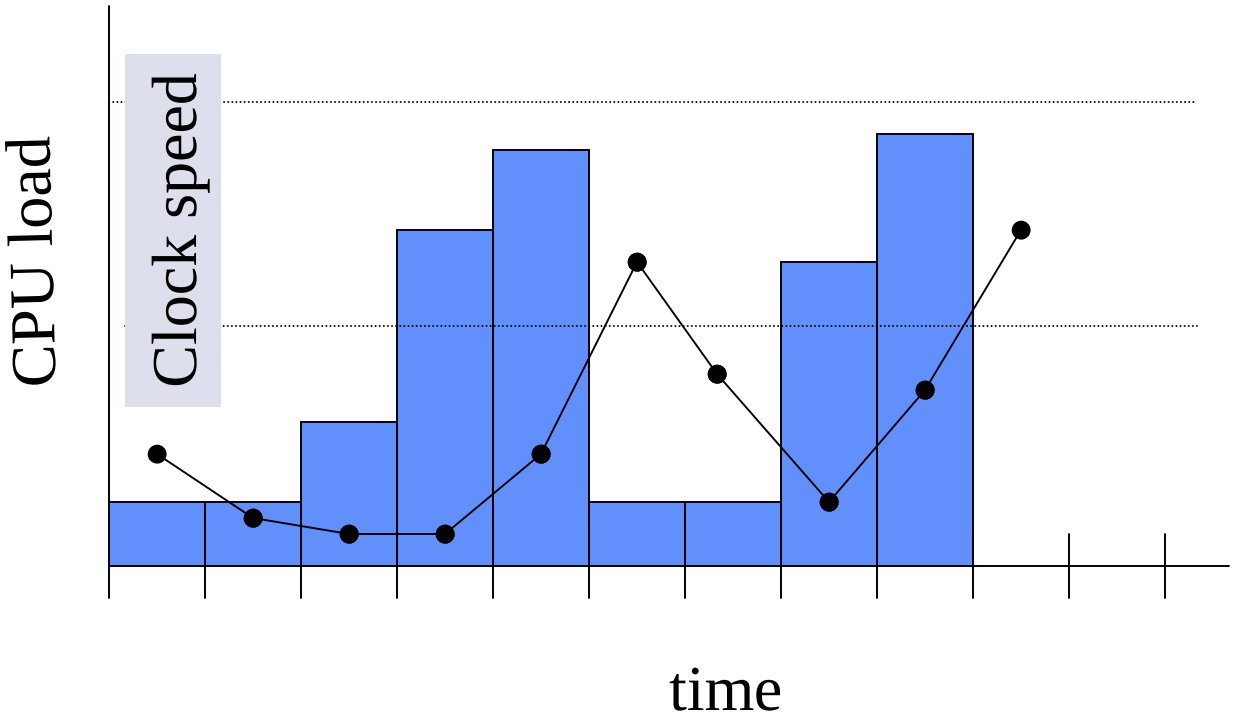
Intuitive goal - fill “soft idle” times with slow computation

Background Work in DVS

- Interval scheduling
  - Based on observed processor utilization
  - “general purpose” -- no deadlines assumed by the system
  - Predicting patterns of behavior to squeeze out idle times.
- Worst-case real-time schedulers (Earliest Deadline First)
  - Stretch the work to smoothly fill the period without missing deadlines (without inordinate transitioning).

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

Implementation of Interval 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
    $W_t = (NW_{t-1} + U_{t-1}) / (N+1)$
- Inelastic performance constraints – don’t want to allow user to see any performance degradation

Results

- 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 can either block or spin)
  - NEED application-supplied information
- Simple averaging shows asymmetric behavior – clock rate drops faster than ramps up
- AVG_N may not stabilize on the “right” clock speed - Oscillations

Earliest Deadline First

Dynamic algorithm

Priorities are assigned to tasks according to the deadlines of their current request

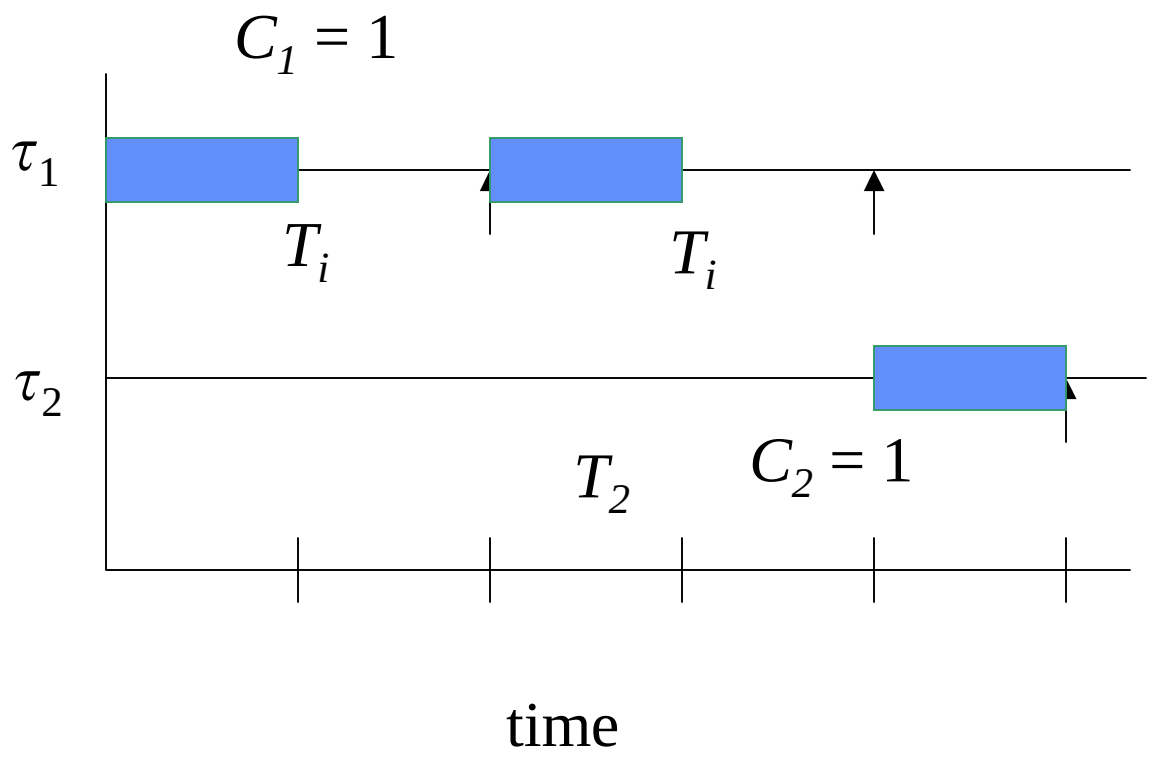
With EDF there is no idle time prior to an overflow

For a given set of *m* tasks, EDF is feasible iff

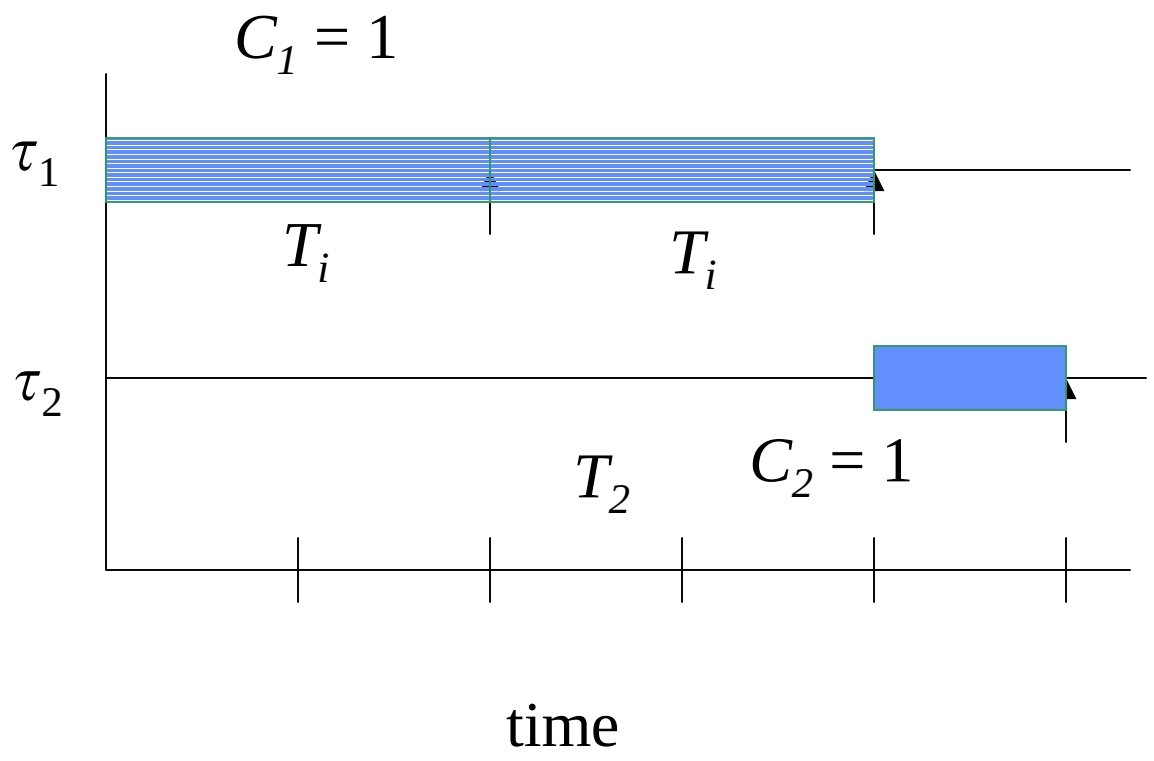
$$C_1/T_1 + C_2/T_2 + \ldots + C_m/T_m \leq 1$$

If a set of tasks can be scheduled by any algorithm, it can be scheduled by EDF

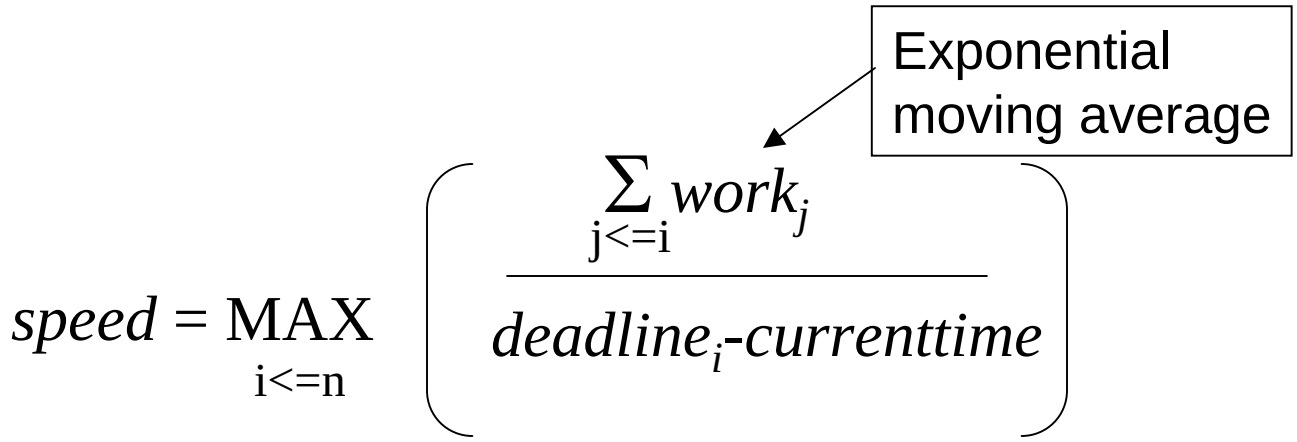
Intuition



Intuition



EDF-based DVS Algorithm



Sort in EDF order

Invoked when thread added or removed or deadline reached

Includes non-runnable in scheduling decision

Relationships

Power (watts) = Voltage (volts) * Current (amps)

Power (watts) = Energy (Joules) / Time (sec)

Energy (Joules) = Power (watts) * Time (sec)

Energy (Joules) = Voltage (volts) * Charge (coulombs)

Current (amps) = Voltage (volts) / Resistance (ohms)