Processor Voltage Scheduling for Real-Time Tasks with Non-Preemptible Sections

F.Zhang, Samuel T. Chanson
Hanyang univ
DMC Lab ,2006/09/20
Jongmin, Kim

Contents

Background and Motivation

- Background
  - Increasing mobile computing.
  - Text/voice => multimedia
  - Energy concern
    - Power consumption
    - Limited battery capacity

- Motivation
  - How to save energy in mobile devices
  - PVS (Processor Voltage Scaling) : an effective way to reduce energy dissipation by reducing the processor speed
  - Energy aware computing
    - Disk
    - Display
    - CPU => up to 25% of system power

Goals and Contributions

- Goals
  - Reduce energy consumption used by CPU.
  - Voltage scheduling for real-time periodic tasks with non-preemptible blocking sections.

- Contributions
  - Reduce processor energy consumption by up to 80 percent over the static speed scheme

Related Work

- PVS (Processor Voltage Scaling)
  - Save energy by reducing the supply voltage as well as the operating frequency

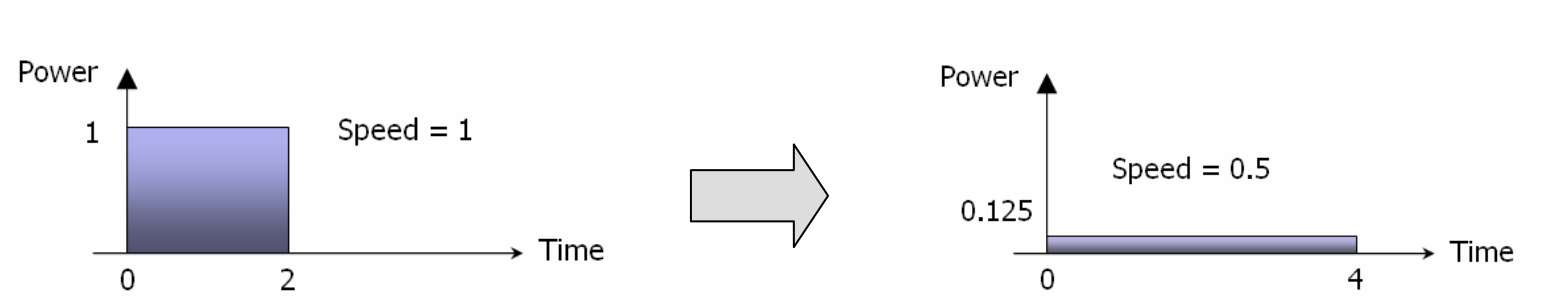
$$P = C \cdot f \cdot V_s^2 \quad \Rightarrow P \propto V_s^3$$

    - P : power consumption rate
    - Vs : supply voltage
    - f: Clock frequency /processor speed
    - C : switched capacitance

  - decreasing the processor speed by lowering the supply voltage , the power consumption will be reduced

Related Work

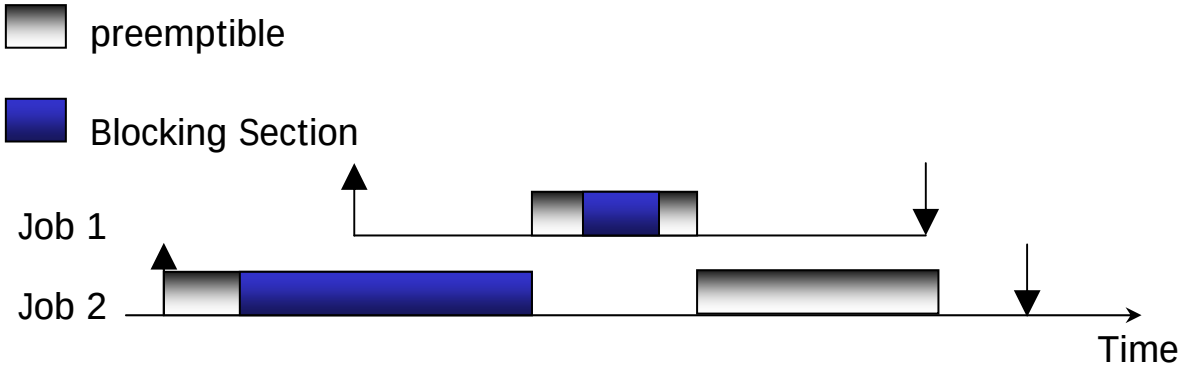
- examples
  - *c* can not control using software.
  - Fully preemptible or non-preemtible.



Saved 75% energy consumption

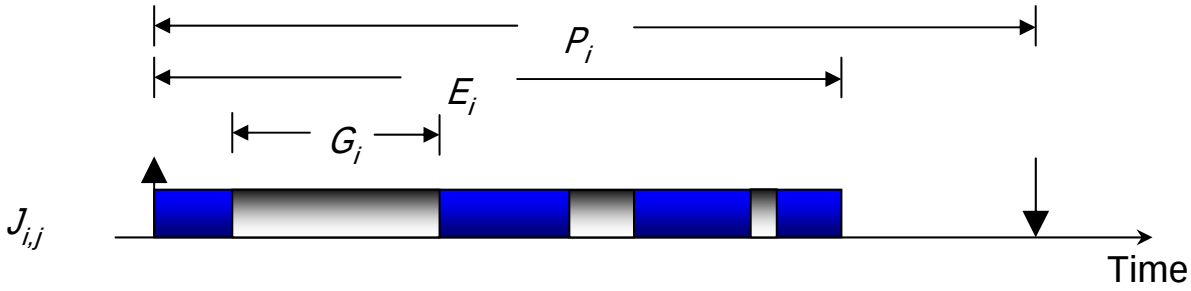
System Model

- Blocking Section
  - Non-preemptible region in preemptible jobs
    - System call with non-preemptive kernel
    - Running in critical section
    - Holding certain types of resources…
  - A high priority job may be blocked by a low priority job running in its blocking section



System Model

- 2.1 Task Model
  - Periodic Task Model
    - Ai : time the task is first released
    - Di : relative deadline of the task
    - Pi : period of the task
    - Ei : worst-case execution time (WCET) of any jog in the task
    - Gi : max length of Blocking Sections



- Processor Model
  - Vmax , Vmin : processor voltage.
  - Smax , Smin : processor speed
    - P = K*Vs3, K= constant

Static Blocking-aware Voltage Scheduling

- Stack resource policy (SRP)
  - A job is not allowed to start unless it has acquired all the resources needed
  - Feasibility condition (for EDF):

$$\forall k,\ 1 \le k \le n, \sum_{i=1}^{k} \frac{E_i}{D_i} + \frac{B_k}{D_k} \le 1$$

  - B_k : Maximum length task T_k can be blocked
- Static speed voltage scaling (for EDF)
  - Speed changes on task arrival or task completion

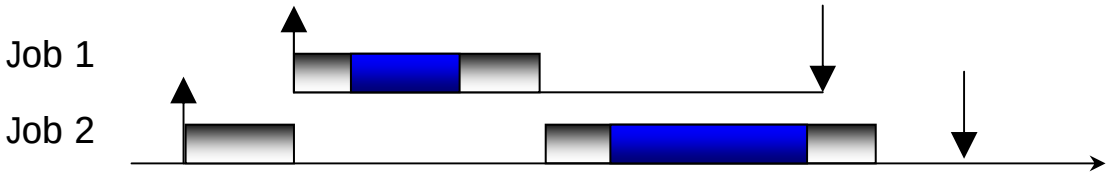
$$\forall k,\ 1 \le k \le n, \sum_{i=1}^{k} \frac{E_i}{P_i} + \frac{\max\{G_j \mid P_j > P_k\}}{P_k} \le H$$

H : Static speed ($H \le 1$)

Dynamic Blocking-aware Voltage Scheduling

- A dual-Speed Algorithm
  - Minimum static speed for fully preemptible tasks

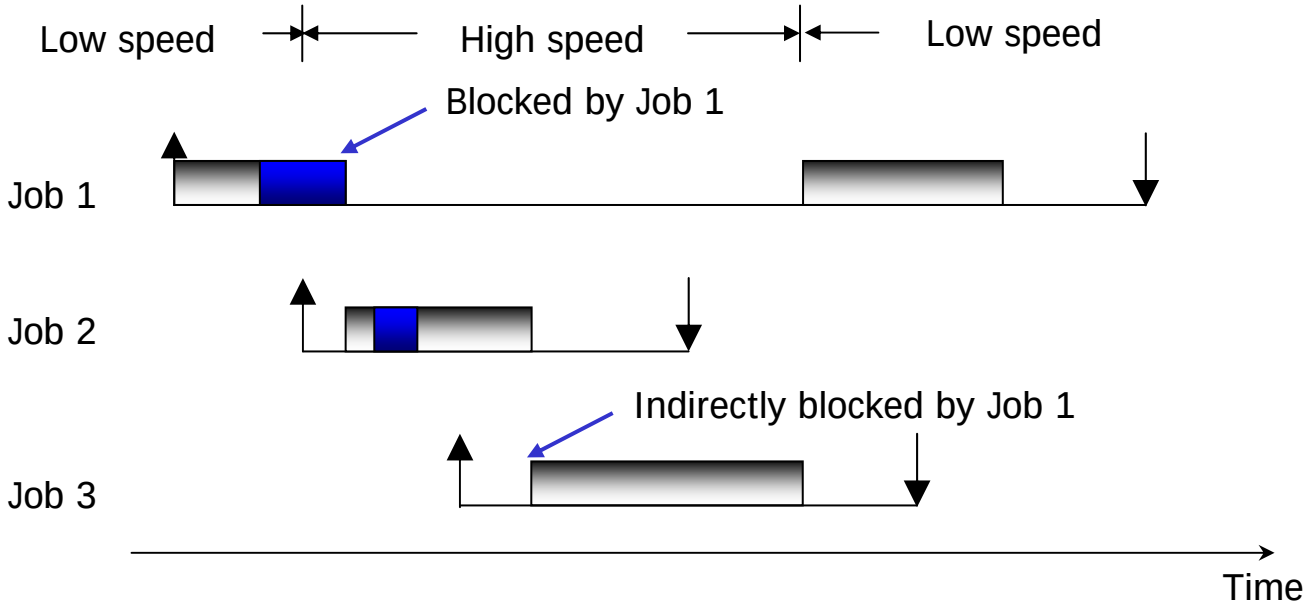
$$\sum_{i=1}^{n} \frac{E_i}{P_i} \le L \qquad L \le H$$



Blocking Section is not H

Dynamic Blocking-aware Voltage Scheduling

- A Dual Speed Algorithms
  - High speed Intervals



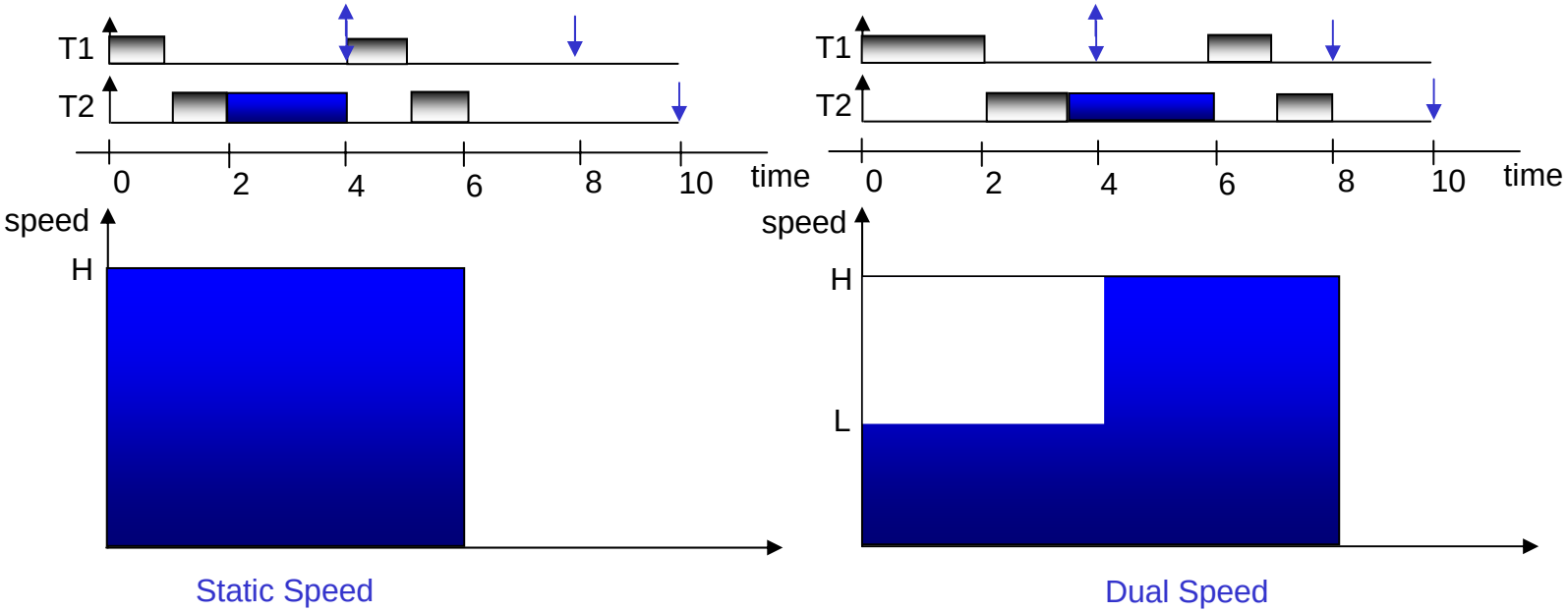
Dynamic Blocking-aware Voltage Scheduling

- Processor starts running at low speed *L*
- Switch to high speed *H* when blocking occurs
- Switch back to *L* if any of the following conditions occurs:
  - Processor becomes idle
  - A job whose deadline is later than or equal to D_b starts execution

D_b: Deadline of the blocking job causing speed changed to *H*

Dynamic Blocking-aware Voltage Scheduling

- 4.2 A Dynamic Reclaiming Algorithm



H = 2L ,Dual speed saves 25%

Dynamic Blocking-aware Voltage Scheduling

- Notation
  - J_i : the current job of task T_i
  - $R_i^r(t)$: the available run time of job J_i at time t
  - $R_i^F(t)$: the run time in the FRT-list that can be used by job J_i at time T
  - $E_i^r(t)$: the worst-case residue execution time of job J_i under the maximum speed S_{max} at time t
- *Run Time*
  - Base speed: simulates the speed that a job will take under dual-speed algorithm
  - Run time assigned to job so it can complete at base speed

$$run\_time = \frac{WCET}{base\_speed}$$

Dynamic Blocking-aware Voltage Scheduling

- DSDR (Dual Speed Dynamic Reclaiming)
  - *n periodic tasks with blocking sections are sorted by their periods. They can be feasibly scheduled by dual-speed EDF with high speed H and low speed L if*

$$\forall k,\ 1 \le k \le n, \sum_{i=1}^{k} \frac{E_i}{P_i} + \frac{\max\{G_j \mid P_j > P_k\}}{P_k} \le H$$

  - *and*

$$\sum_{i=1}^{n} \frac{E_i}{P_i} \le L$$

DSDR is used with high speed H and low speed L, the run time of a job is always depleted at or before the job's deadline.

Dynamic Blocking-aware Voltage Scheduling

- DSDR Characteristics.
  - Running at utilization speed *L* keeps processor fully utilized (i.e., no idle time)
    - Each job can run for $WCET_i/L$ time
  - Jobs may not run for so long under dual-speed
    - Jobs affected by blocking run at high speed *H*
    - Jobs do not always use up their *WCET* s
  - Subsequent jobs can run longer at lower speed utilizing the slack => Energy is saved.

Dynamic Blocking-aware Voltage Scheduling

- Free run-time list (FRT): collecting residue time from
  - Job does not use up its run time gives up all its residue
  - Job with base speed *H* contributes (WCET/L – WCET/H) to FRT with deadline D_b
    D_b: Deadline of the blocking job causing base speed changed to *H*
- *Actual Process Speed*

$$speed = \frac{E^r}{R^r + R^f}$$

Dynamic Blocking-aware Voltage Scheduling

- Algorithms

When a new job (J_i) arrives:
  $E_i^r(t) = E_i$;
  $R_i^r(t) = E_i/L$;
  if Priority(J_i) > Priority(current job)
    if Preempt_Current_Job() is successful
      Select J_i to run;
    else /* the job is blocked */
      $base\_speed = H$;
      $End\_H = \max(End\_H, d_{current\_job})$;
      Set_Speed(H);
    end if
  end if

When job J_i is selected to run:
  if J_i is executed for the first time && $base\_speed == H$
  /* first run reclamation */
    Insert_To_FRT($R_i^r(t) - E_i/H, End\_H$);
    $R_i^r(t) = E_i/H$;
  end if
  Set_Speed($\frac{E_i^r(t)}{R_i^F(t)+R_i^r(t)}$);
  Execute J_i;

When job J_i completes:
  if $R_i^r(t) > 0$
    Insert_To_FRT($R_i^r(t)$, d_i);
    /*early completion reclamation*/
  end if

When the end of high speed interval is reached:
  $End\_H = -1$;
  $base\_speed = L$;

Experimental Results

- Simulation setup
  - Processor voltage: 0 ~ V_{max} step 0.1V_{max}
  - Task setup: 30 tasks, 10 in each category

Task Type	Short Task	Medium Task	Long Task
Period	20~100ms	100~1000ms	1000~5000ms
WCET	1~20ms	1~100ms	1~1000ms

    - Execution times are scaled so that total task utilization does not exceed the specified utilization factor
    - Blocking factor is the proportion of a blocking section in a job's execution time
  - Blocking probability: probability that a job contains a blocking section
  - Slack factor: actual execution time of a job is varied between its WCET and (1 – slack_factor)·WCET

Experimental Results

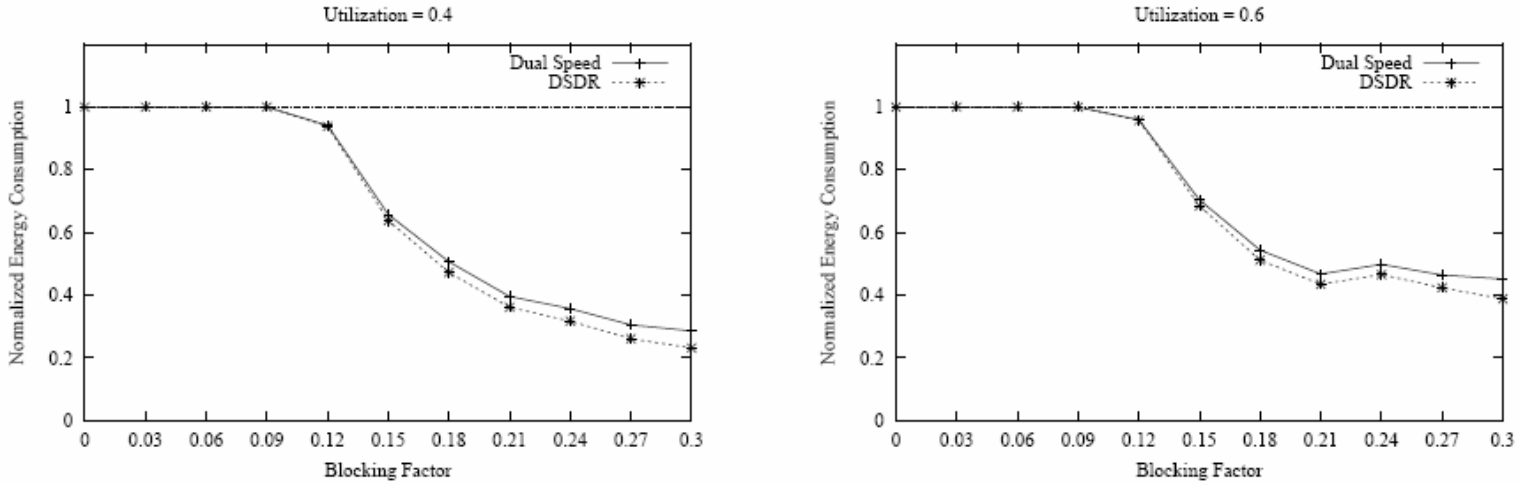


Figure 5. Normalized energy consumption with varying blocking factor.

Experimental Results

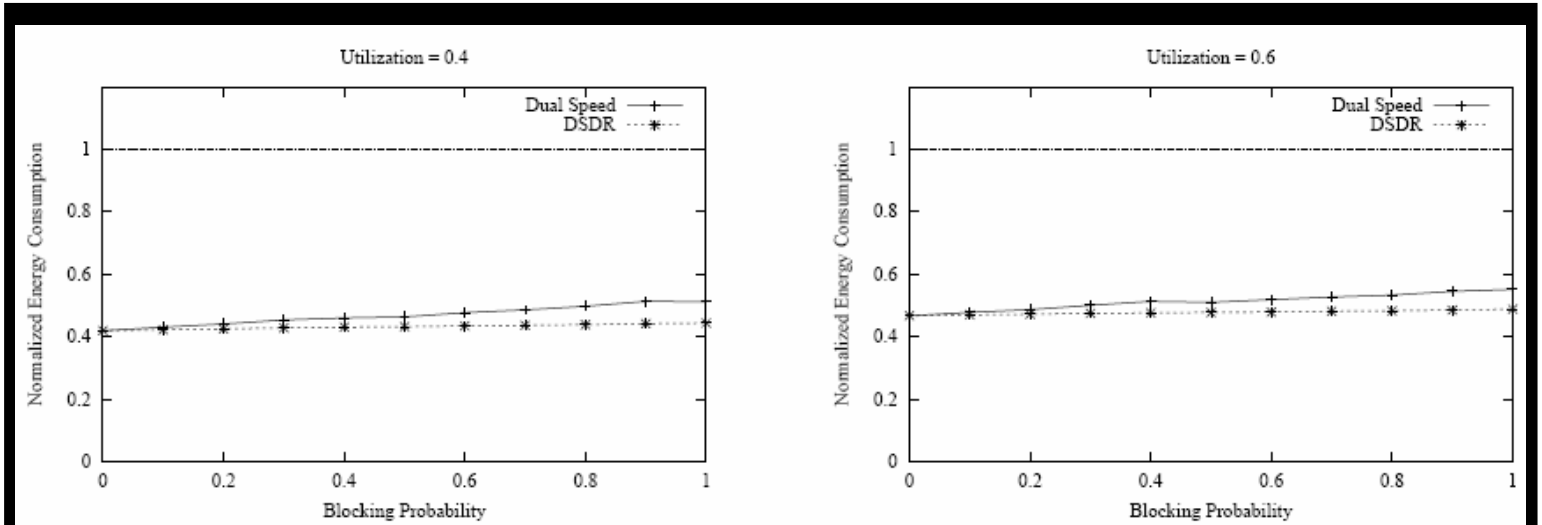


Figure 6. Normalized energy consumption with varying blocking probability.

Conclusion

- Summary
  - PVS (Processor Voltage Scaling) : use voltage speed.
  - SRP (Stack Resource Policy) : Base of Static Voltage Scaling.
  - DPDS (Dual Speed Dynamic Reclaiming) : unused run time and redistributed it to jobs that are able to make use of it

  - Thank You~!