



## Dynamic Voltage Scaling



# Why Consider Energy?

---

Low end:

- Battery-operated devices (laptops, phones, wireless sensors, ...)
- Processor speed grows faster than battery capacity: energy becomes a bottleneck

High end:

- Cost of energy is increasing: The energy bill is the second highest operational expense of data centers (Google, HP, IBM, ...)



# Power of Computation

---

- Terminology
  - $R$  : Power spent on computation
  - $V$  : Processor voltage
  - $f$  : Processor clock frequency
  - $R_0$  : Leakage power
- Power spent on computation is:
  - $R = k_v V^2 f + R_0$   
where  $k_v$  is a constant



# Energy of Computation

---

- Power spent on computation is:
  - $R = k_v V^2 f + R_0$
- Consider a task of length  $C$  clock cycles and a processor operating at frequency  $f$
- The execution time is  $t = C/f$
- Energy spent is:
  - $E = R t = (k_v V^2 f + R_0)(C/f)$



# Reducing Processor Frequency

---

- Power spent on computation is:
  - $R = k_v V^2 f + R_0$
- Energy spent is:
  - $E = R t = (k_v V^2 f + R_0)(C/f)$
- Question:
  - Does it make sense to operate the processor at a reduced speed to save energy? Why or why not?



# Reducing Processor Frequency

## Good or Bad?

---

- Does it make sense to operate the processor at a reduced speed to save energy? Why or why not?

Possible Answer:

$$E = R t = (k_v V^2 f + R_0)(C/f) = k_v V^2 C + R_0 C/f$$

- Conclusion:  $E$  is minimum when  $f$  is maximum.
  - Operate at top speed
- Is this really true? What are the underlying assumptions?

# Dynamic Voltage Scaling (DVS):

## Reducing Voltage and Frequency

---

- Processor voltage can be decreased if clock frequency is decreased
  - Voltage and frequency can be decreased roughly proportionally.
  - In this case (where  $V \sim f$ ):

$$R = k_f f^3 + R_0$$

$$E = (k_f f^3 + R_0)(C/f) = k_f f^2 C + R_0 C/f$$

# Dynamic Voltage Scaling (DVS):

## Reducing Voltage and Frequency

- Processor voltage can be decreased if clock frequency is decreased
  - Voltage and frequency can be decreased roughly proportionally.

$$R = k_f f^3 + R_0$$

$$E = (k_f f^3 + R_0)(C/f) = k_f f^2 C + R_0 C/f$$

- Question: Does reducing frequency (and voltage) increase or decrease total energy spend on a task?



# Dynamic Voltage Scaling (DVS):

## The Critical Frequency

---

- There exists a minimum frequency below which no energy savings are achieved

$$E = k_f f^2 C + R_0 C / f$$

$$dE/df = 2k_f f C - R_0 C / f^2 = 0$$

$$f = \sqrt[3]{\frac{R_0}{2k_f}}$$

# Dynamic Voltage Scaling (DVS)



## Example:

---

- A processor uses  $10\text{mW}$  when running at full speed and  $3\text{mW}$  when running at half speed. How much energy is saved, if any, at half speed? (If energy, in fact, increases, use a negative sign to indicate “negative savings”)



# DVS Algorithm 1: Static Voltage Scaling

---

1. Calculate the critical frequency
2. Calculate the minimum frequency at which the task set remains schedulable
  - Example: If EDF is used and the utilization is 60% at the maximum frequency  $f_{max}$ , then the frequency can be decreased to  $0.6 f_{max}$ .
3. Let  $f_{opt}$  be the larger of the above two
4. Operate the system at the smallest frequency at or above  $f_{opt}$ .



# Pros and Cons

---

Pro:

- Very simple and can be done offline

Con:

- Algorithm is based on worst-case estimates of task execution length
  - What if a task finishes early?



# DVS Algorithm 2: Cycle-conserving DVS

---

- What if a task finishes early?
  - Re-compute the utilization based on the reduced execution time.
  - Calculate the minimum frequency at which the task set is schedulable using the new utilization.
  - Update task execution times to the WCET when new invocations are released.



# Pros and Cons

---

Pro:

- Very simple (updates utilization only)

Con:

- Pessimistic – assumes that all future invocations will have the worst-case execution length until shown otherwise

# Practical Consideration:

## Accounting for Off-chip Overhead

- In the preceding discussion, we assumed that task execution *time* at frequency  $f$  is  $C/f$ , where  $C$  is the total cycles needed
- In reality some cycles are lost waiting for memory access and I/O (Off-chip cycles).
  - Let the number of CPU cycles used be  $C_{cpu}$  and the time spent off-chip be  $C_{off-chip}$
  - Execution time at frequency  $f$  is given by

$$C_{cpu}/f + C_{off-chip}$$