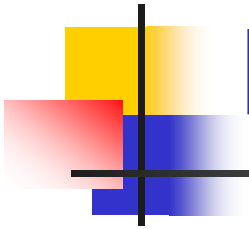


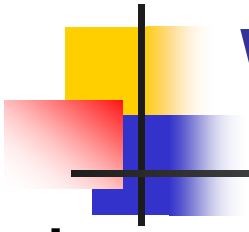
Energy

Dynamic Voltage Scaling



Reminder

- If you are taking this class for 4 credits, please send me your topic and group members by Feb 29th
- Topics voted in class
  - In-depth look into Android OS
  - OS-level virtualization: Containers, Jails, etc.
  - Bios and Bootloaders
  - A comparison of Windows and Linux
  - OS Architecture: Monolithic, Microkernels, Exokernels and more
  - Supercomputer Operating Systems
  - Implementation of the Network Stack (and new paradigms)



Why Consider Energy?

Low end:

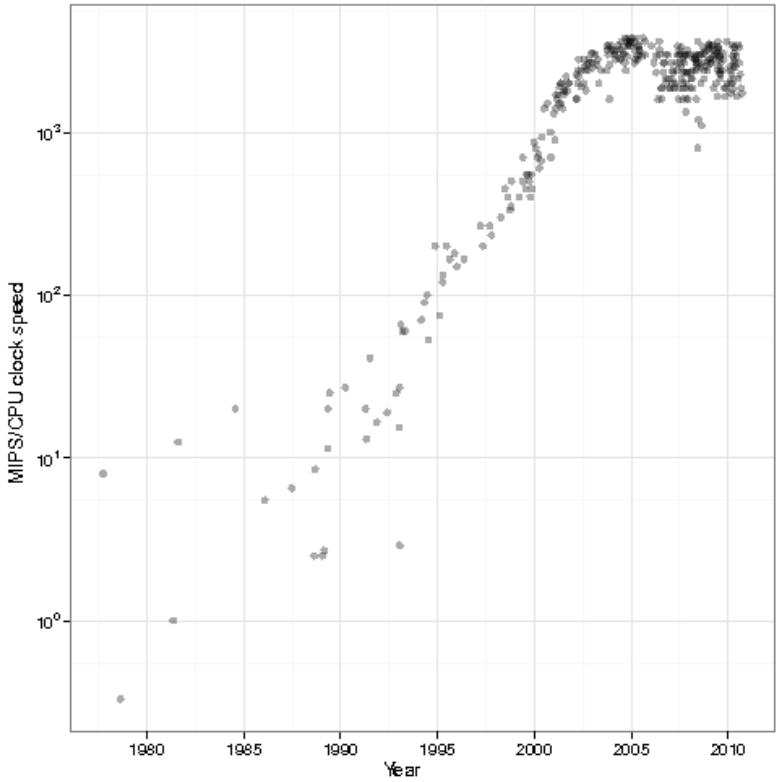
- Battery-operated devices (laptops, phones, wireless sensors, …)
- Processor speed grows faster than battery capacity: energy becomes a bottleneck

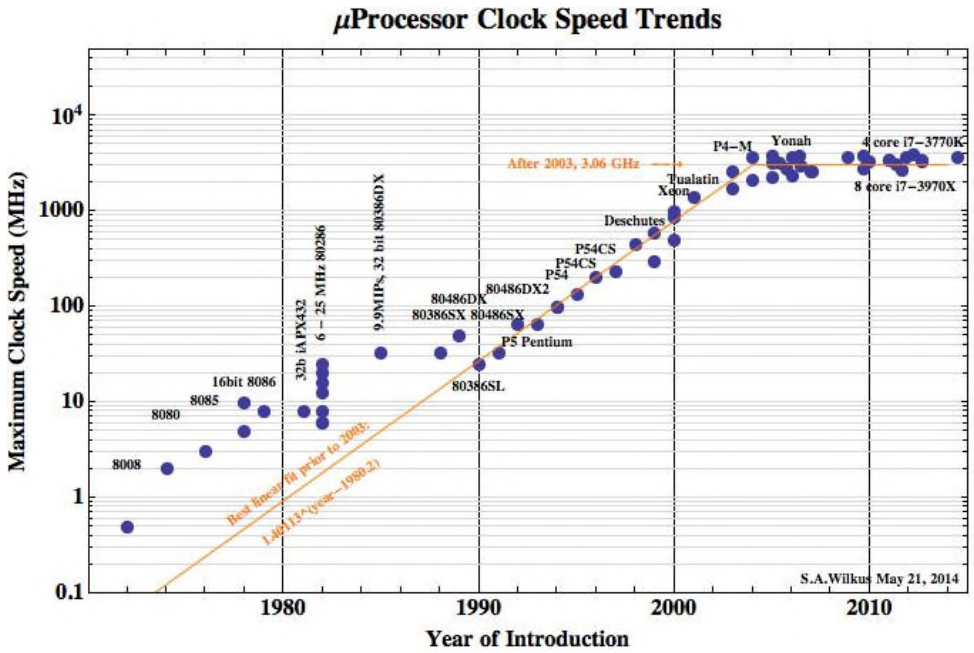
High end:

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

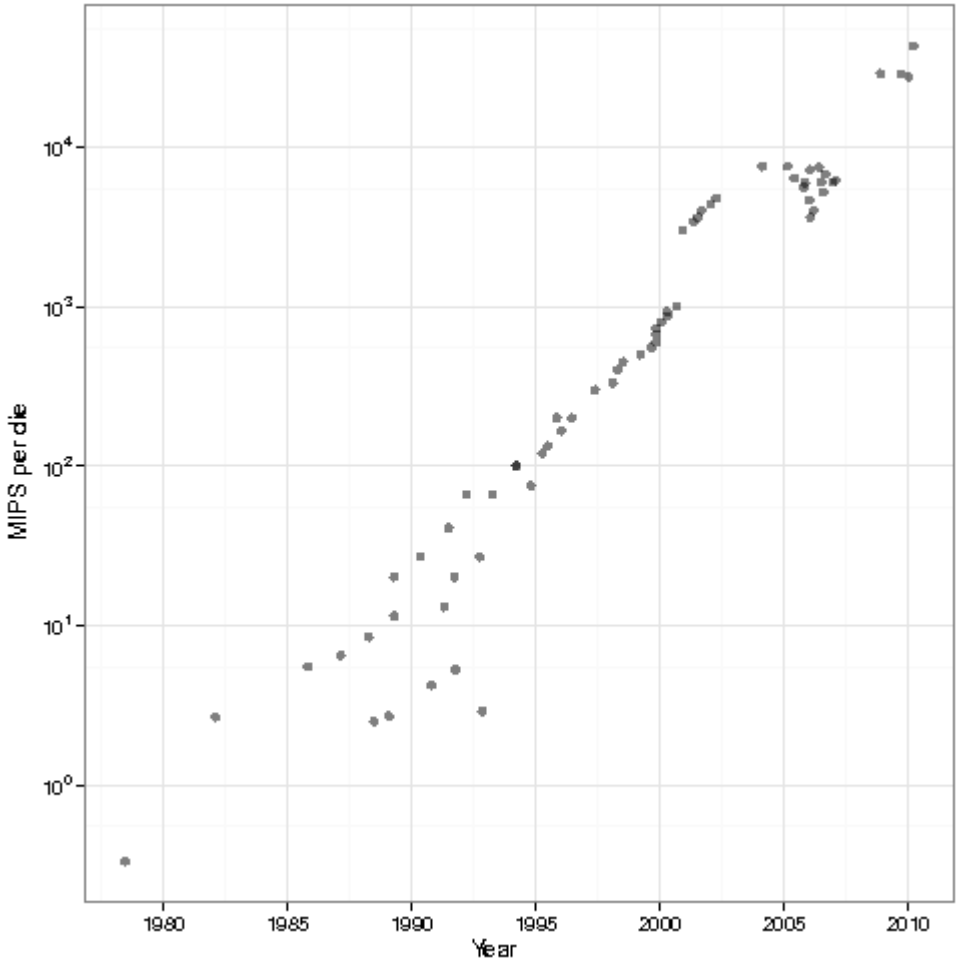
Microprocessor Clock Speed

- Moore's Law (1980-2005)
- Question: Why did the speed curve saturate





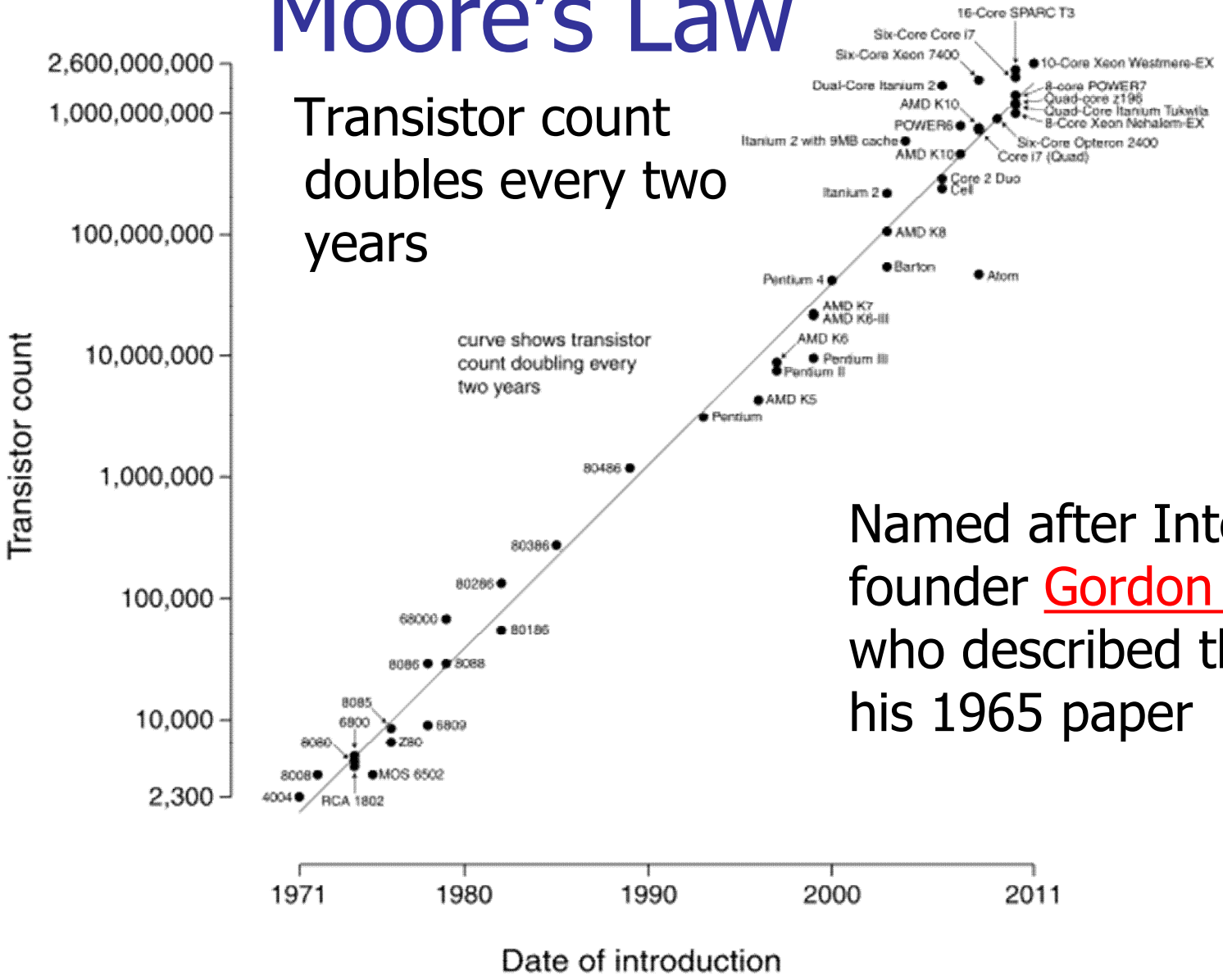
Computational Power (per Die)



- Note the exponential rise in power consumption
- Question: how come it does not saturate?

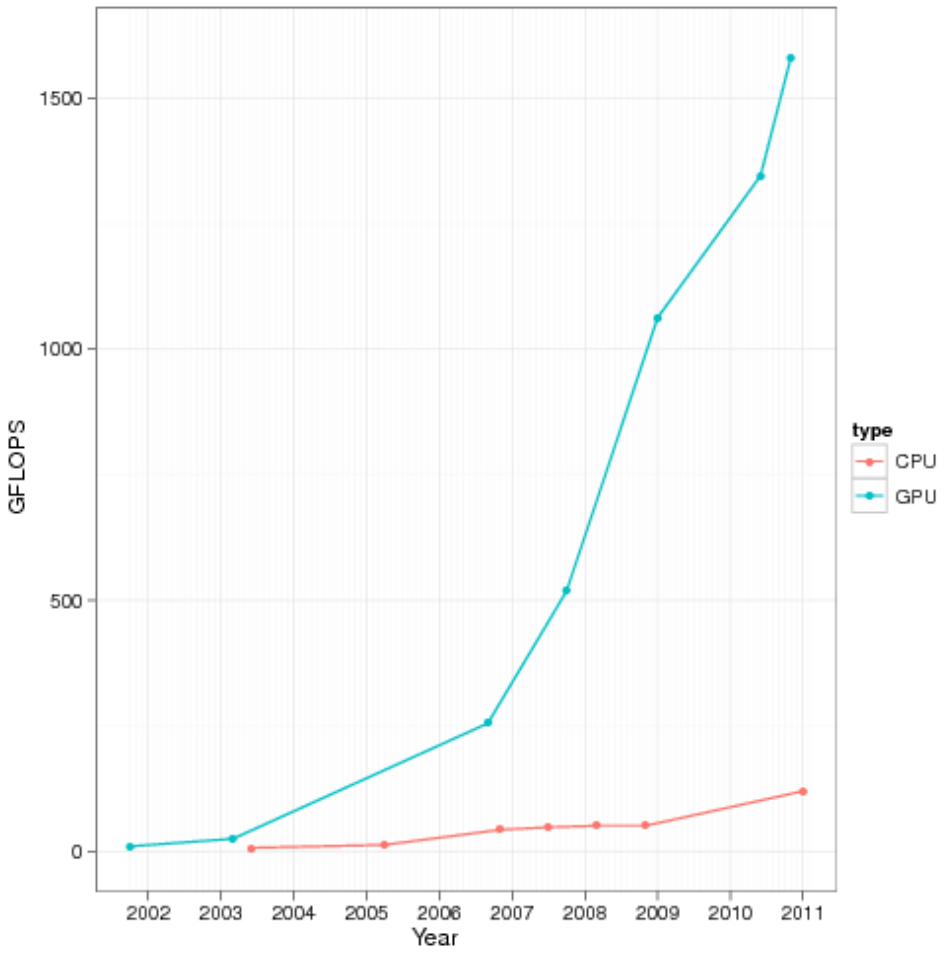
Moore's Law

Transistor count doubles every two years



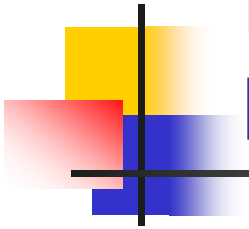
Named after Intel co-founder Gordon E. Moore, who described the trend in his 1965 paper

CPUs versus GPUs

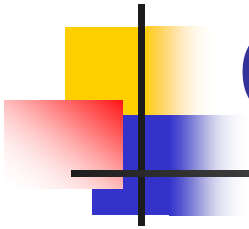


- GPUs are much better at floating point operations

Advanced Configuration and Power Interface (ACPI)
Part of UEFI since 2013



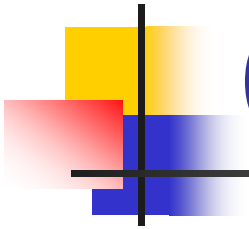
- Defines different power saving states in a platform-independent manner
- The standard was originally developed by Intel, Microsoft, and Toshiba (in 1996), then later joined by HP, and Phoenix.
- The latest version is "Revision 6" published in April 2015.



Global States

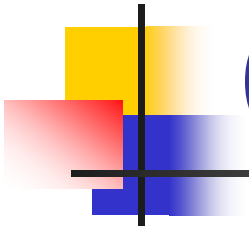
- **G0**: *working*
- **G1**: *Sleeping* and *hibernation* (several degrees available)
- **G2**:, *Soft Off*: almost the same as G3 *Mechanical Off*, except that the power supply still supplies power, at a minimum, to the power button to allow wakeup. A full reboot is required.
- **G3**, *Mechanical Off*: The computer's power has been totally removed via a mechanical switch (as on the rear of a PSU).

Processor "Sleep" States (C-states)

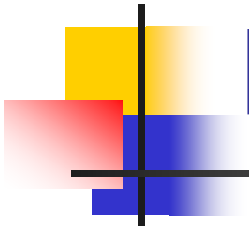


- **C0**: is the operating state.
- **C1** (often known as *Halt*): is a state where the processor is not executing instructions, but can return to an executing state instantaneously. All ACPI-conformant processors must support this power state.
- **C2** (often known as *Stop-Clock*): is a state where the processor maintains all software-visible state, but may take longer to wake up. This processor state is optional.
- **C3** (often known as *Sleep*) is a state where the processor does not need to keep its cache, but maintains other state. This processor state is optional.

Processor Performance States (P-States)



- **P0** max power and frequency
- **P1** less than P0, voltage/frequency scaled
- **P2** less than P1, voltage/frequency scaled
- …
- **Pn** less than *P(n-1)*, voltage/frequency scaled



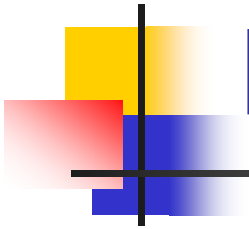
Power of Computation

- 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 piece of computation 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):

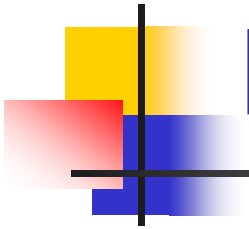
The Critical Frequency

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

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

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

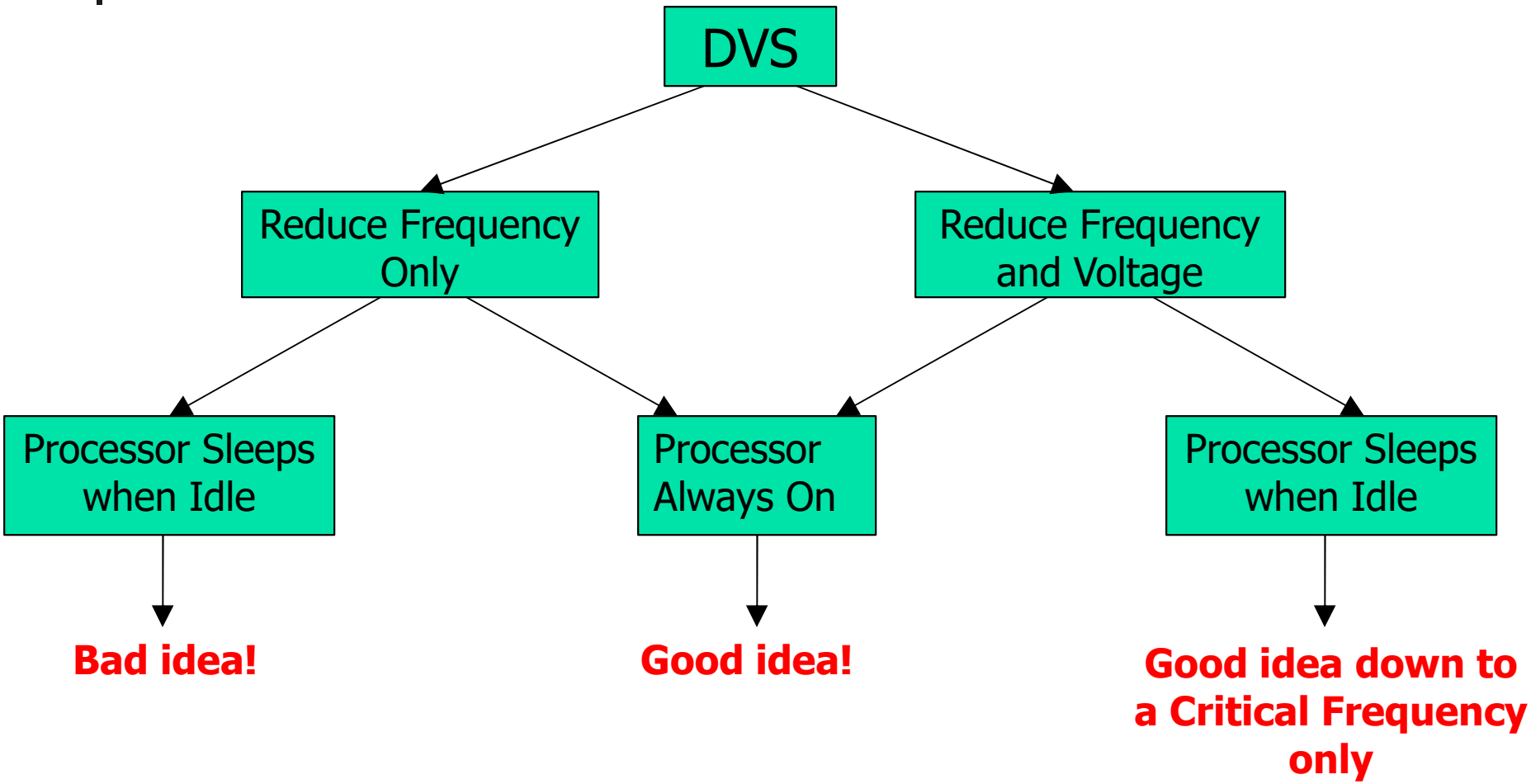
$$f = \sqrt[3]{\frac{R_0}{2k_f}}$$



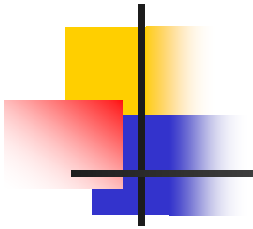
Linux CPUFreq Governor

- Linux defines multiple DVS modes (called CPUfreq "governors"):
  - Performance (highest frequency)
  - Powersave (lowest frequency)
  - Userspace ("root" user controls frequency)
  - OnDemand (adaptively change frequency depending on load)

Recap


DVS
Reduce Frequency Only
Reduce Frequency and Voltage
Processor Sleeps when Idle
Processor Always On
Processor Sleeps when Idle
Bad idea!
Good idea!
Good idea down to a Critical Frequency only

Dynamic Voltage Scaling (DVS)



Example:

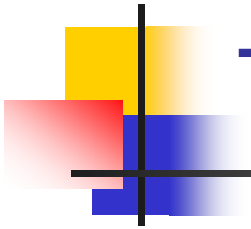
- A processor uses $10\mathrm{mW}$ when running at full speed and $3\mathrm{mW}$ when running at half speed. How much energy is saved, if any, at half speed? (If energy, in fact, increases, use a negative sign to indicate "negative savings")

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 $C_{off\text{-}chip}$
  - Execution time at frequency f is given by

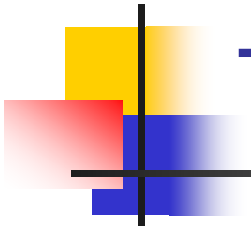
$$C_{cpu}/f + C_{off\text{-}chip}$$

Turning Processors Off
The Cost of Wakeup



- Energy expended on wakeup, E_{wake}
- To sleep or not to sleep?

Turning Processors Off
The Cost of Wakeup



- Energy expended on wakeup, E_{wake}
- To sleep or not to sleep?
  - Not to sleep (for time t):

    $$E_{no\text{-}sleep} = (k_v \, V^2 f + R_0) \, t$$

  - To sleep (for time t) then wake up:

    $$E_{sleep} = P_{sleep} \, t + E_{wake}$$

Turning Processors Off
The Cost of Wakeup

- Energy expended on wakeup, E_{wake}
- To sleep or not to sleep?
  - Not to sleep (for time t):

    $$E_{no\text{-}sleep} = (k_v V^2 f + R_0)\, t$$

  - To sleep (for time t) then wake up:

    $$E_{sleep} = P_{sleep}\, t + E_{wake}$$

  - To save energy by sleeping: $E_{sleep} < E_{no\text{-}sleep}$

    $$t > \frac{E_{wake}}{k_v V^2 f + R_0 - P_{sleep}}$$

Turning Processors Off
The Cost of Wakeup

- Energy expended on wakeup, E_{wake}
- To sleep or not to sleep?
  - Not to sleep (for time t):

$$E_{no\text{-}sleep} = (k_v V^2 f + R_0)\, t$$

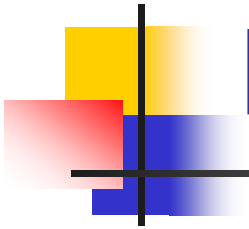
  - To sleep (for time t) then wake up:

$$E_{sleep} = P_{sleep}\, t + E_{wake}$$

  - To save energy by sleeping: $E_{sleep} < E_{no\text{-}sleep}$

$$t > \frac{E_{wake}}{k_v V^2 f + R_0 - P_{sleep}}$$

Minimum sleep interval

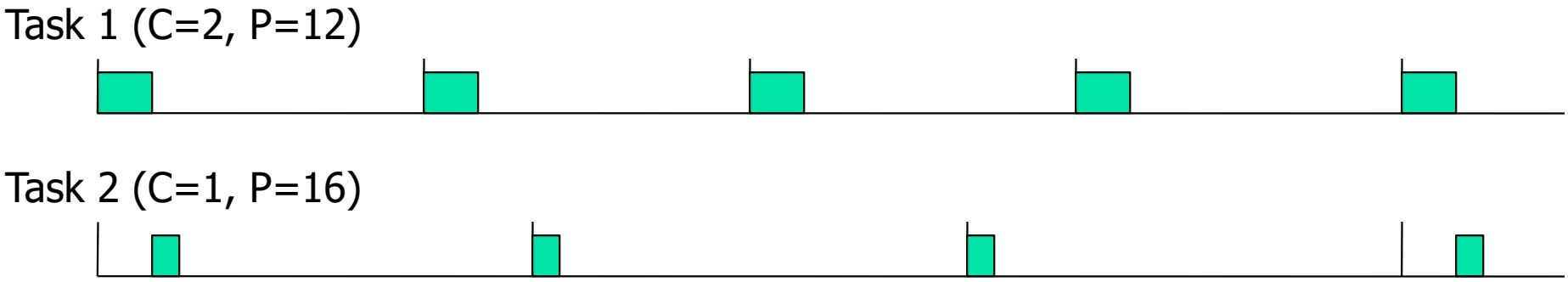


Dynamic Power Management

- DPM refers to turning devices off (or putting them in deep sleep modes)
- Device wakeup has a cost that imposes a minimum sleep interval (a breakeven time)
- DPM must maximize power savings due to sleep while maintaining schedulability

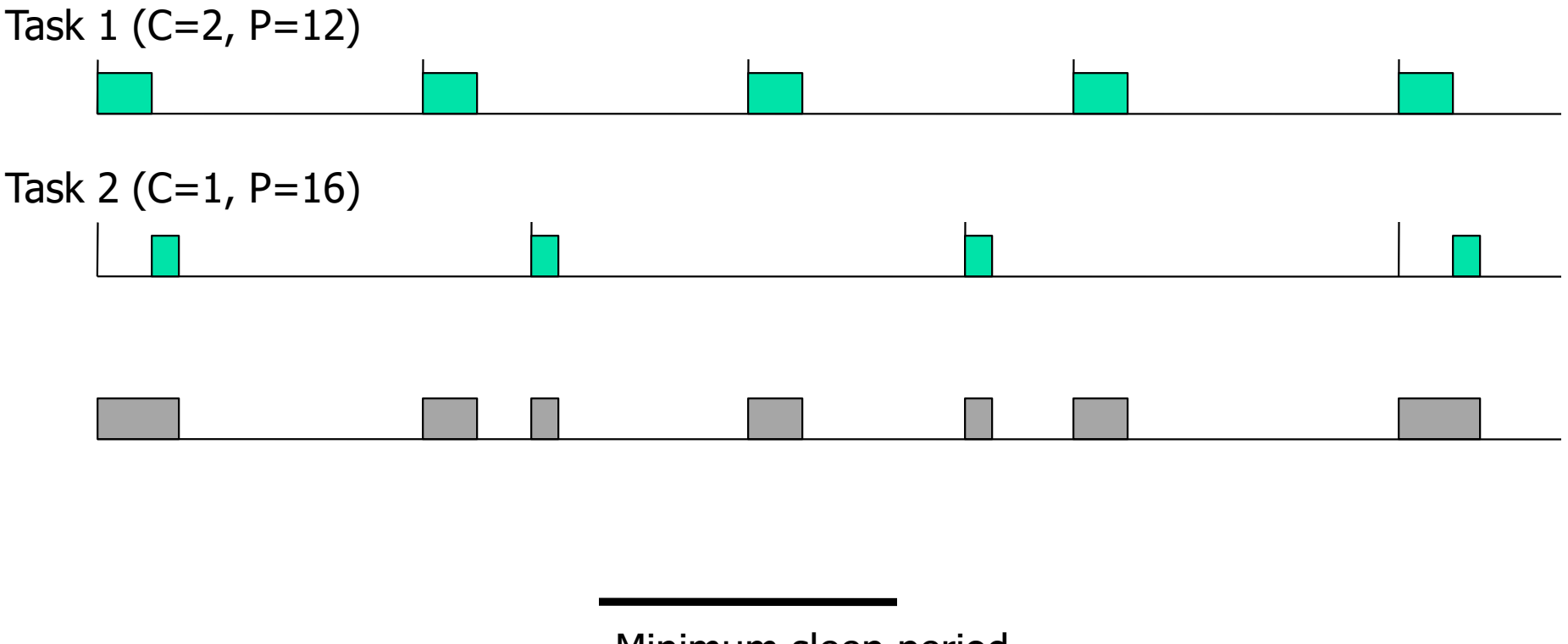
DPM and the Problem with Work-conserving Scheduling

- Example:



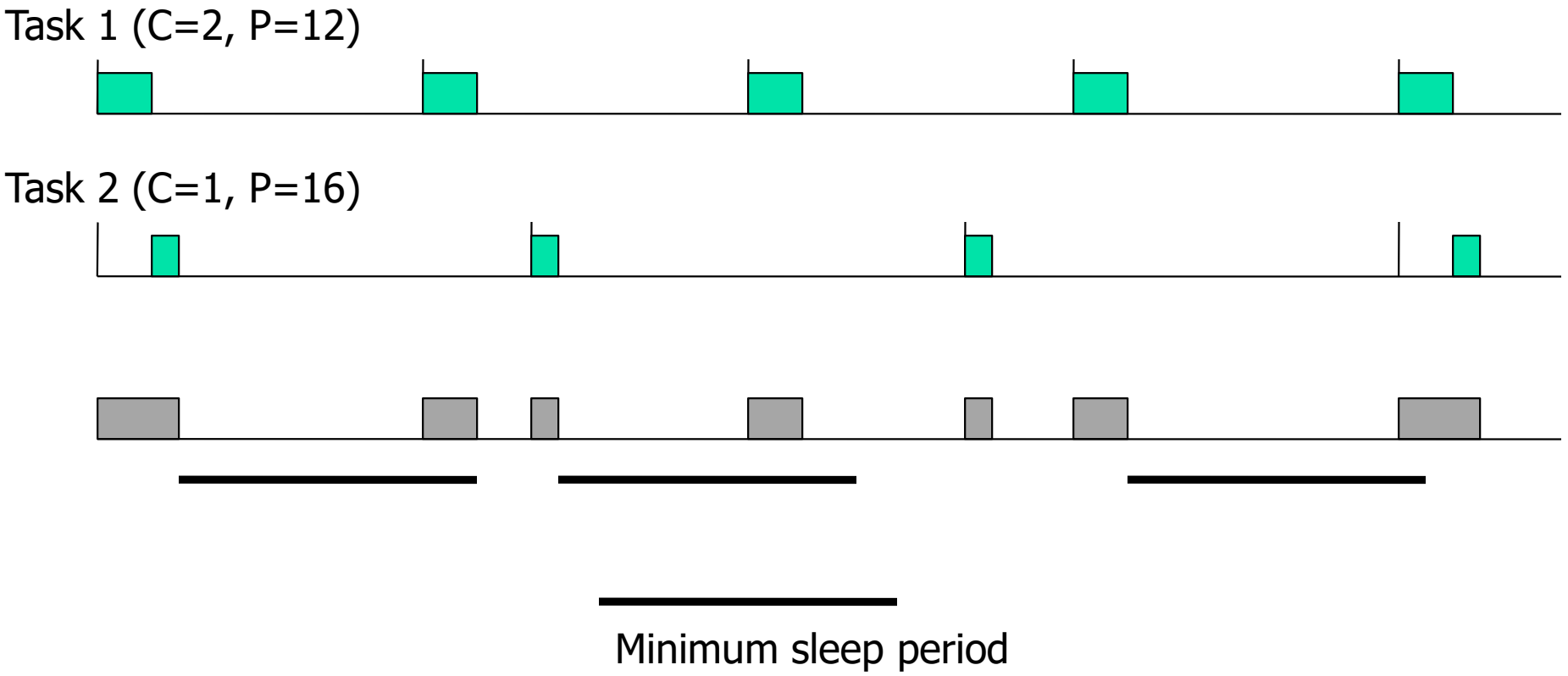
DPM and the Problem with Work-conserving Scheduling

- Example:



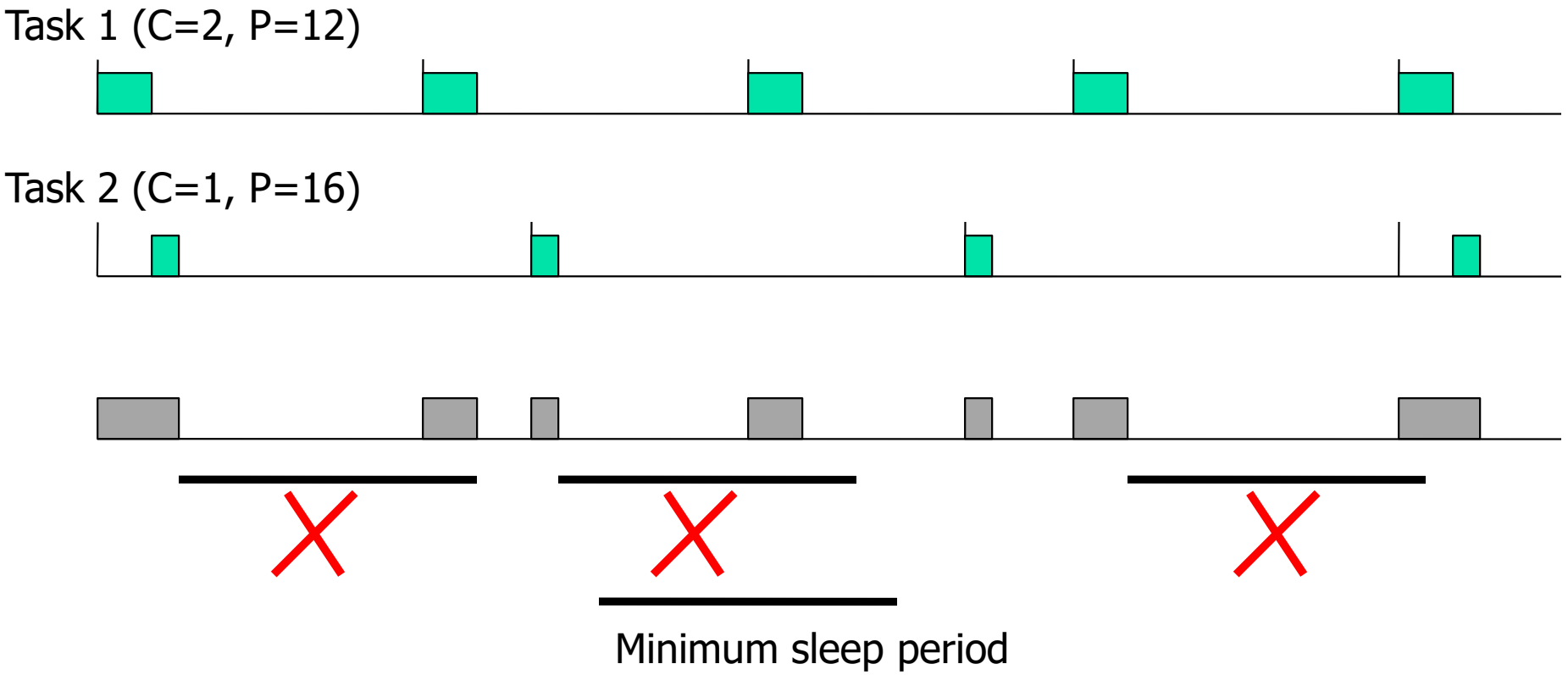
DPM and the Problem with Work-conserving Scheduling

- Example:



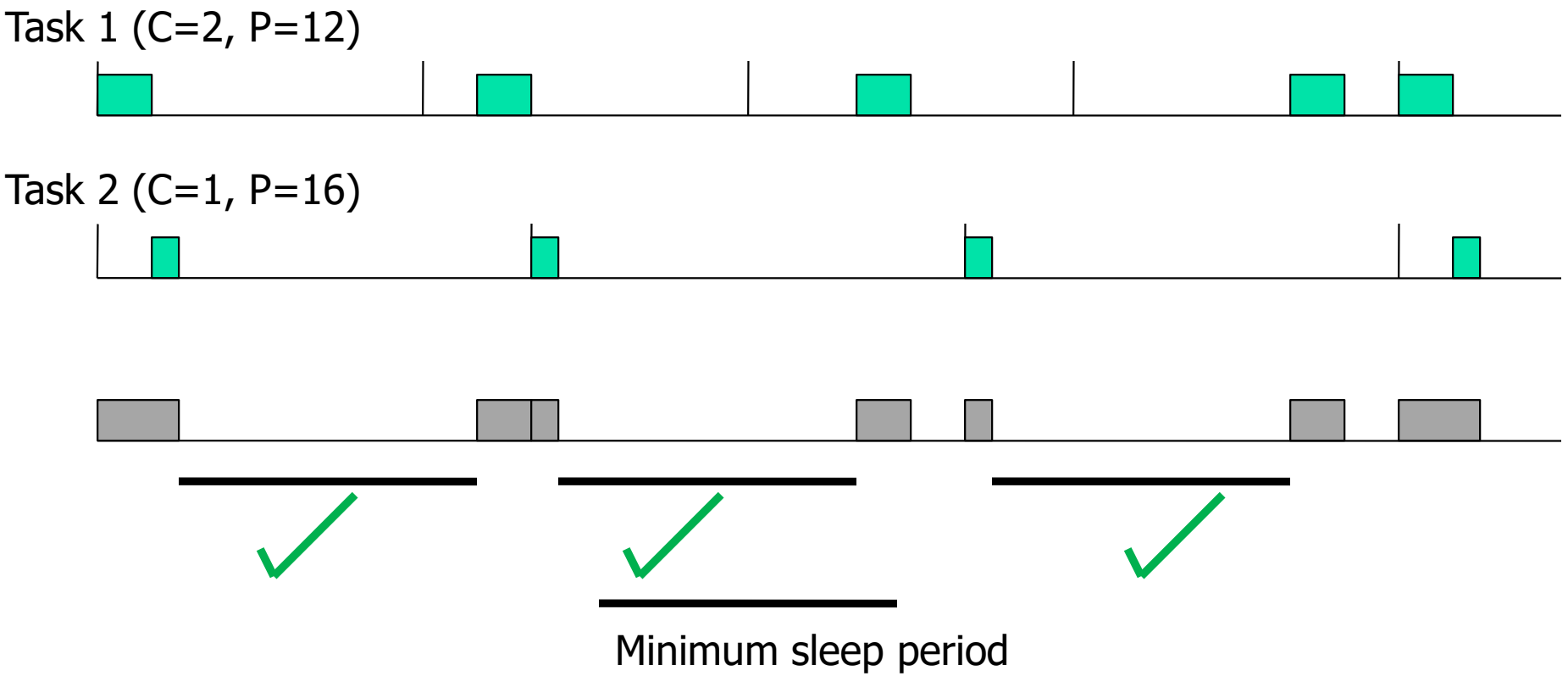
DPM and the Problem with Work-conserving Scheduling

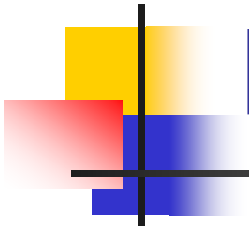
- No opportunity to sleep ☹



DPM and the Problem with Work-conserving Scheduling

- Must batch! ☺





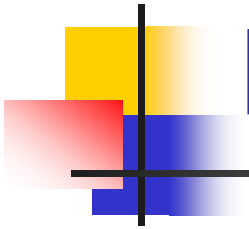
DVS on Multiple Processors

- From the perspective of minimizing energy, is it always a good idea to use up all processors?

How Many Processors to Use?

- Consider using one processor at frequency f versus two at frequency $f/2$
- Case 1: Total power for one processor
  - $k_f f^3 + R_0$
- Case 2: Total power for two processors
  - $2\{k_f (f/2)^3 + R_0\} = k_f f^3/4 + 2R_0$

How Many Processors to Use?



- Consider using one processor at frequency f versus two at frequency $f/2$
- Case 1: Total power for one processor
  - $k_f f^3 + R_0$
- Case 2: Total power for two processors
  - $2\{k_f (f/2)^3 + R_0\} = k_f f^3/4 + 2R_0$
- The general case: n processors
  - $n\{k_f (f/n)^3 + R_0\} = k_f f^3/n^2 + nR_0$

How Many Processors to Use?

- The general case: n processors
  - $Power = n \{k_f (f/n)^3 + R_0\} = k_f f^3/n^2 + n R_0$
  - $dPower/dn = -2 k_f f^3/n^3 + R_0 = 0$

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

How Many Processors to Use?

- The general case: n processors
  - $Power = n \{k_f (f/n)^3 + R_0\} = k_f f^3/n^2 + n R_0$
  - $dPower/dn = -2 k_f f^3/n^3 + R_0 = 0$

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

- What if n is not an integer?