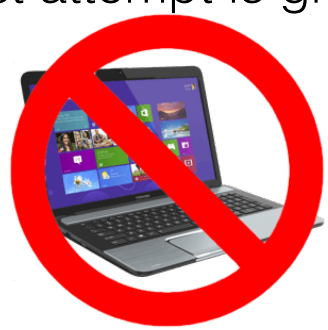
Goals for Today

- Learning Objective:
  - Understand how energy usage informs OS design
- Announcements, etc:
  - MP4 due **May 7th**
    - Deadline provides more time than necessary; wanted to give you flexibility
  - HW1 available! Due **May 4th**
    - Just an “appetizer” for the final exam
    - Multiple attempts allowed, but first attempt is graded



Reminder: Please put away devices at the start of class

CS 423
Operating System Design: Energy + Power Considerations

Professor Adam Bates

Spring 2018

Why care about energy?

Low-end Computers:

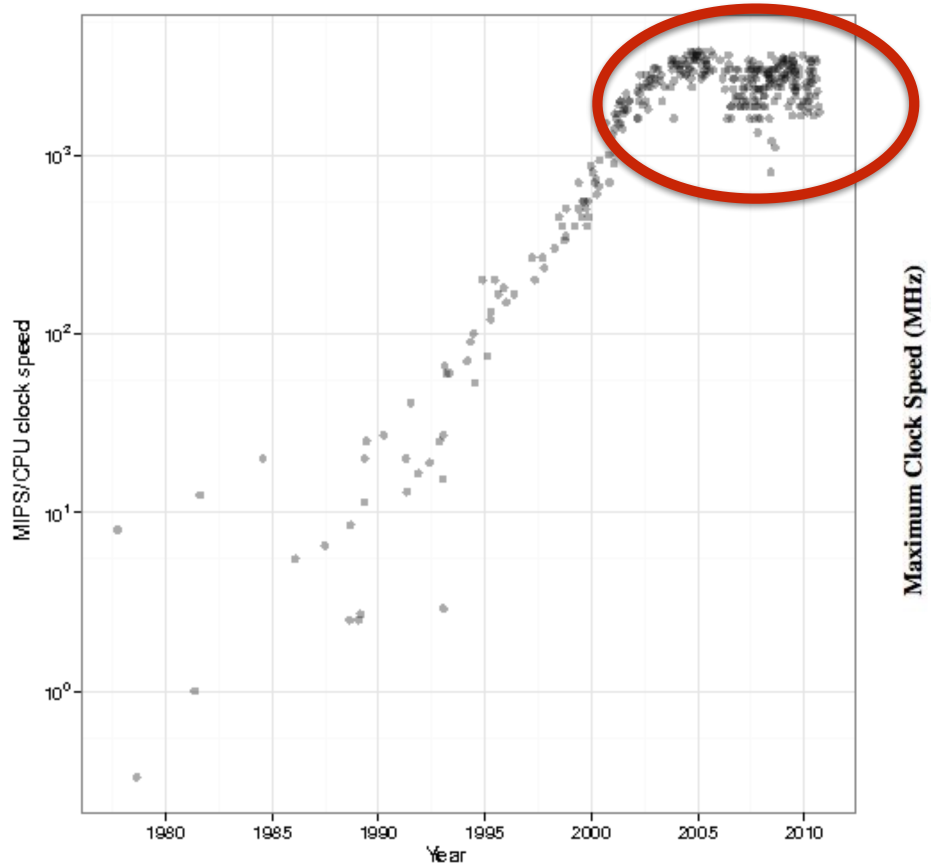
- Resource-constrained battery-operated devices (laptops, phones, wireless sensors, …)
- Processor speed grows faster than battery capacity: energy becomes a bottleneck

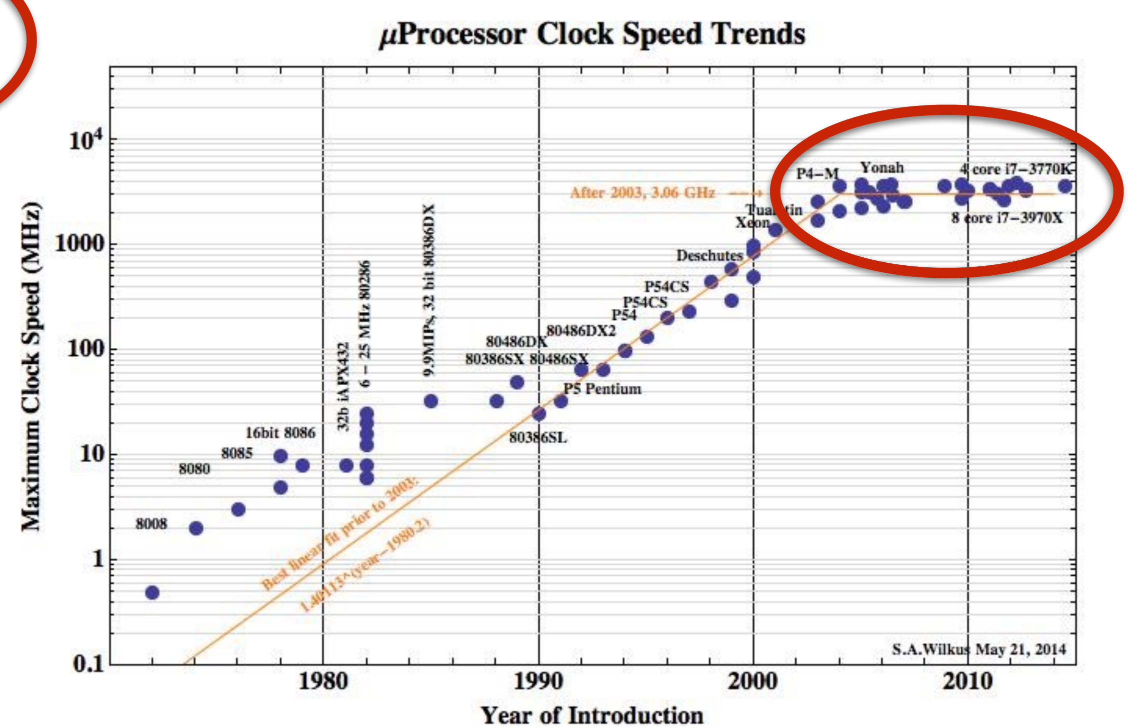
High-end Computers:

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

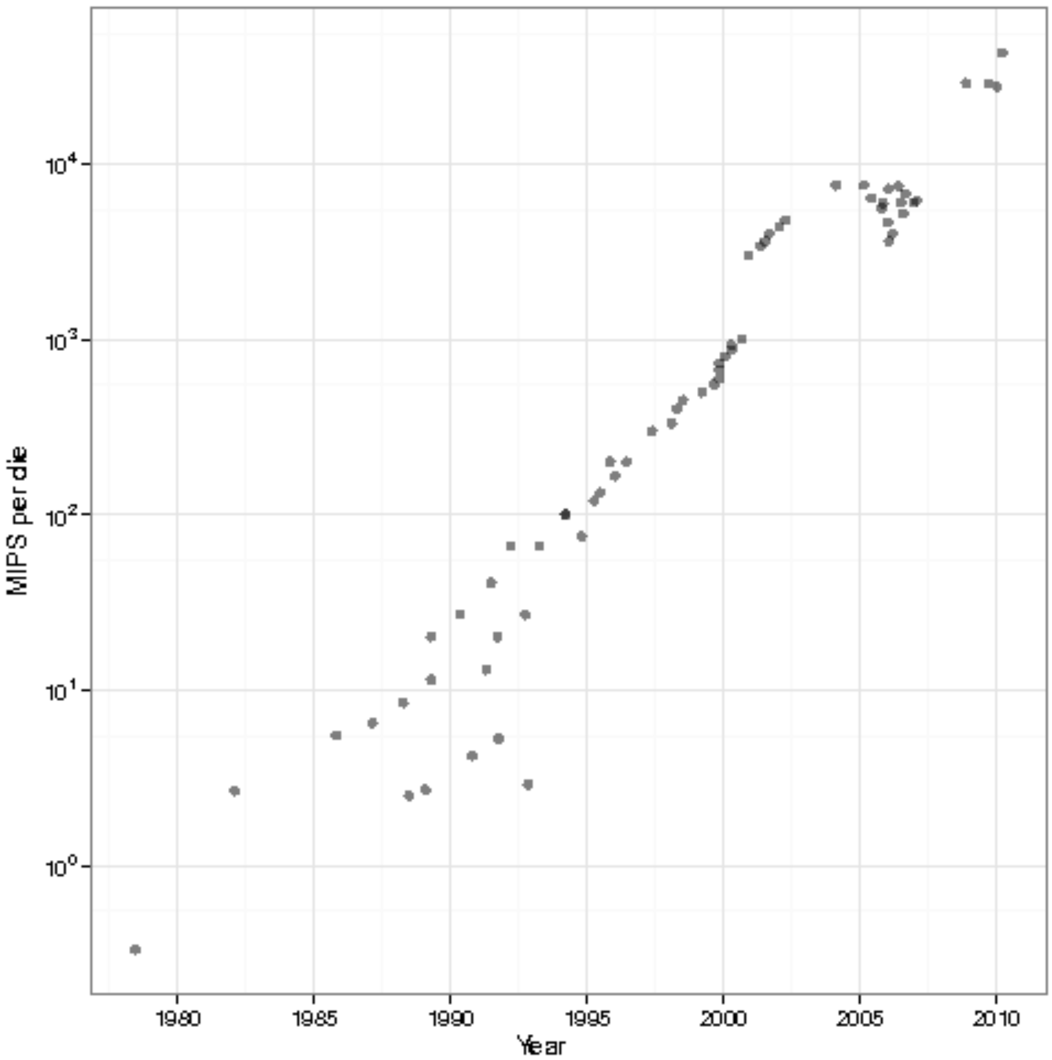
Microprocessor Clock Speed

- Moore's Law (1980-2005)
- Question: Why did the speed curve level off in 2005?





Computational Power (per Die)

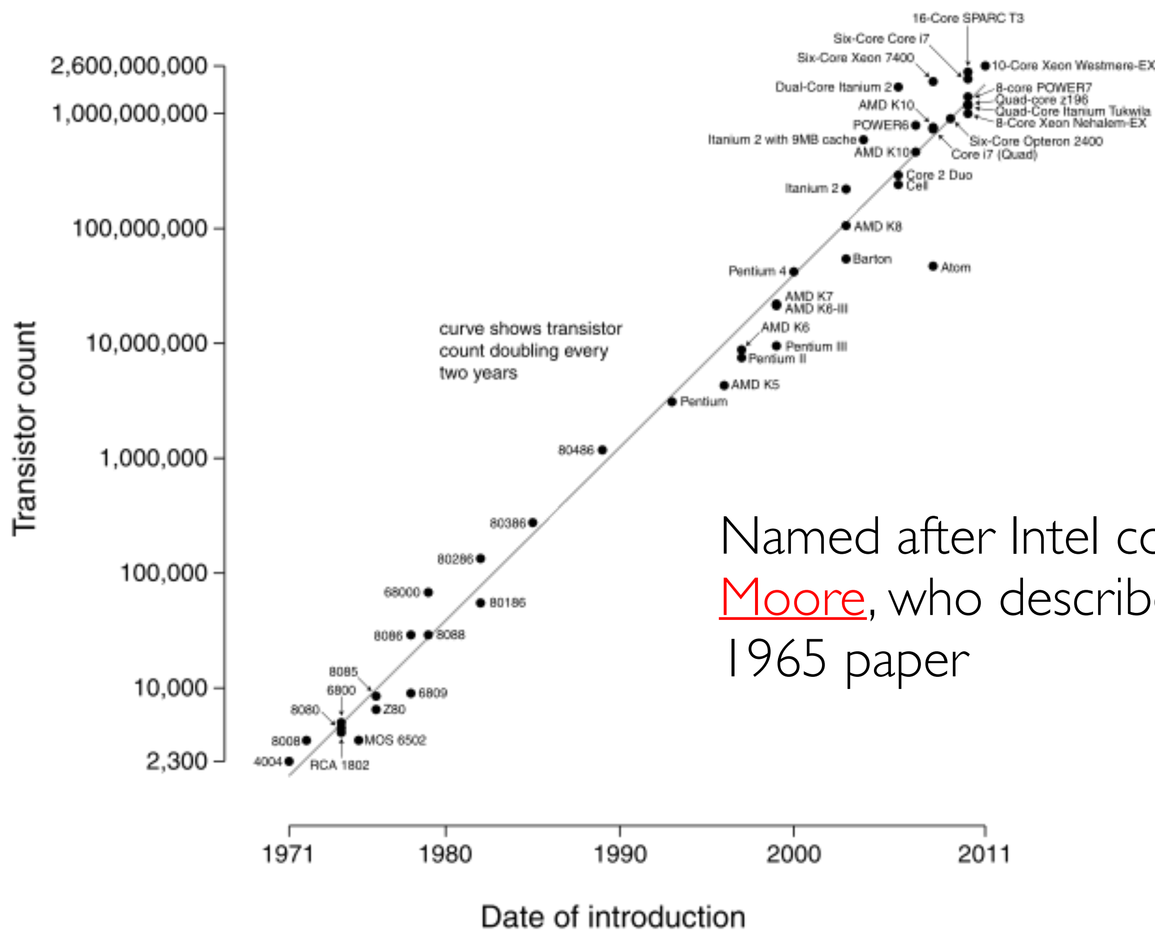


- Note the exponential rise in power consumption
- Question: how come it does not saturate?

Moore's Law

Moore's Law: Transistor count doubles every two years

Microprocessor Transistor Counts 1971-2011 & Moore's Law



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

Advanced Configuration and Power Interface (ACPI)

Part of UEFI since 2013:

- Exposes 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.

ACPI 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).

ACPI 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).

ACPI “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.

ACPI “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.

ACPI Proc 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

Is 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:

- In reality, 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:

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

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

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

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

Dynamic Voltage Scaling (DVS)

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

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

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

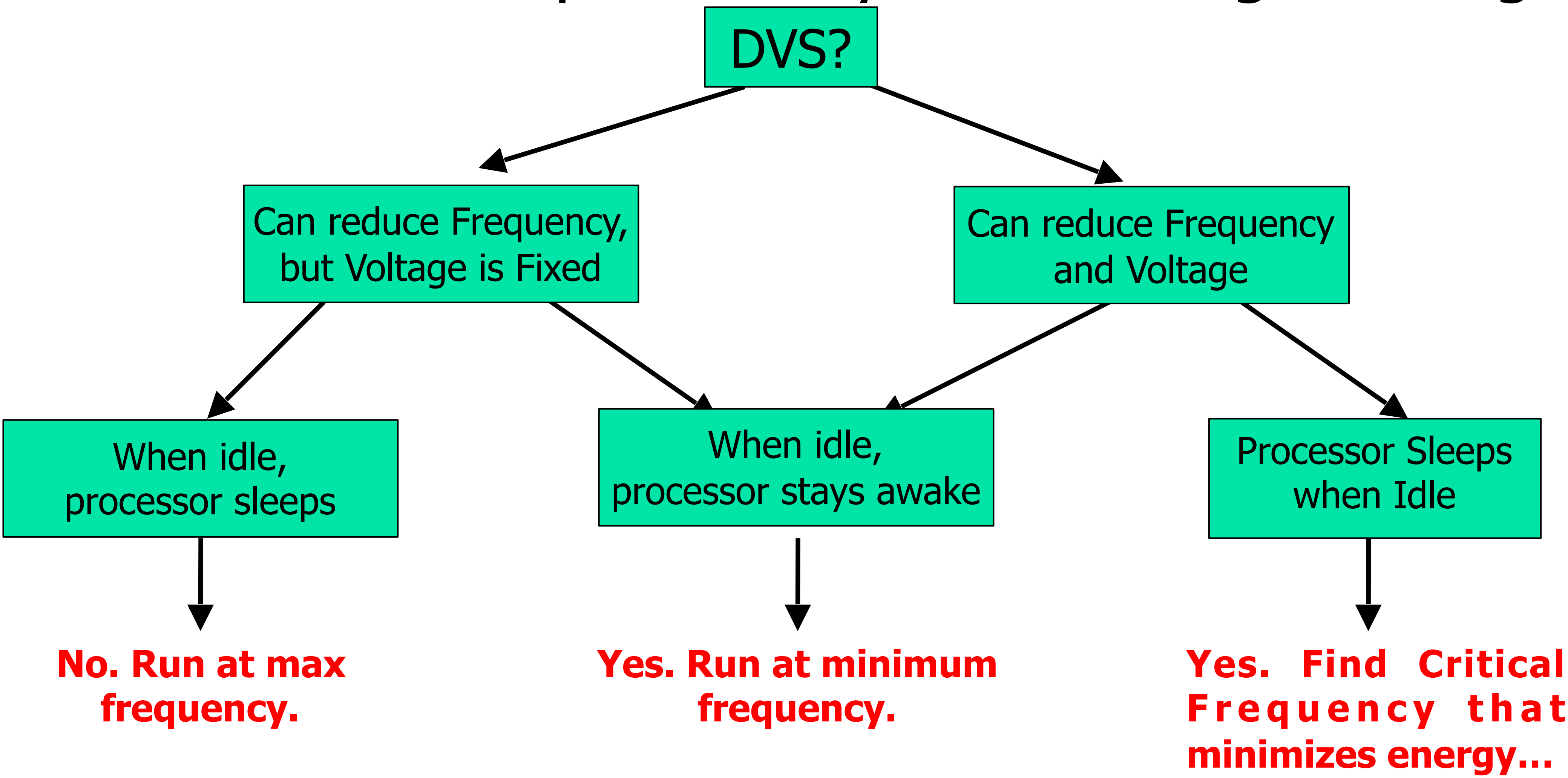
$$f = \sqrt[3]{\frac{R_0}{2k_f}}$$

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)

Dynamic Voltage Scaling (DVS)

When should we perform dynamic voltage scaling?



Accounting for Off-chip

- 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

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

The Cost of Wakeup

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

The Cost of Wakeup

- Turning Processor off…
- 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:

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

The Cost of Wakeup

- Turning Processor off…
- 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: $E_{sleep} < E_{no\text{-}sleep}$

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

The Cost of Wakeup

- Turning Processor off…
- 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 Mgmt

- 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

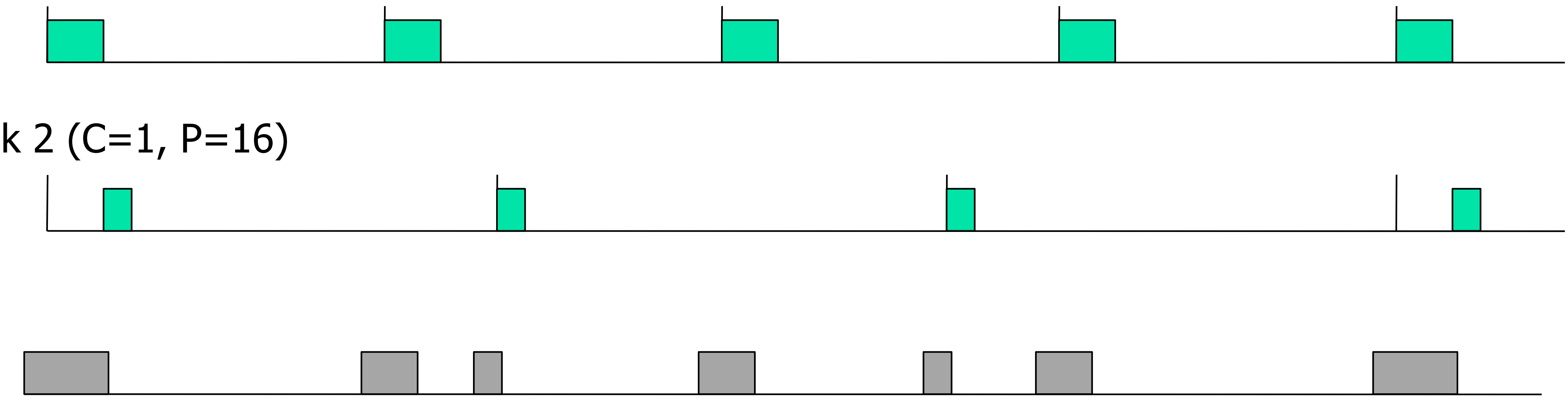
Question

How does dynamic power management affect scheduling?

The Problem with work-conserving scheduling:

Task 1 (C=2, P=12)

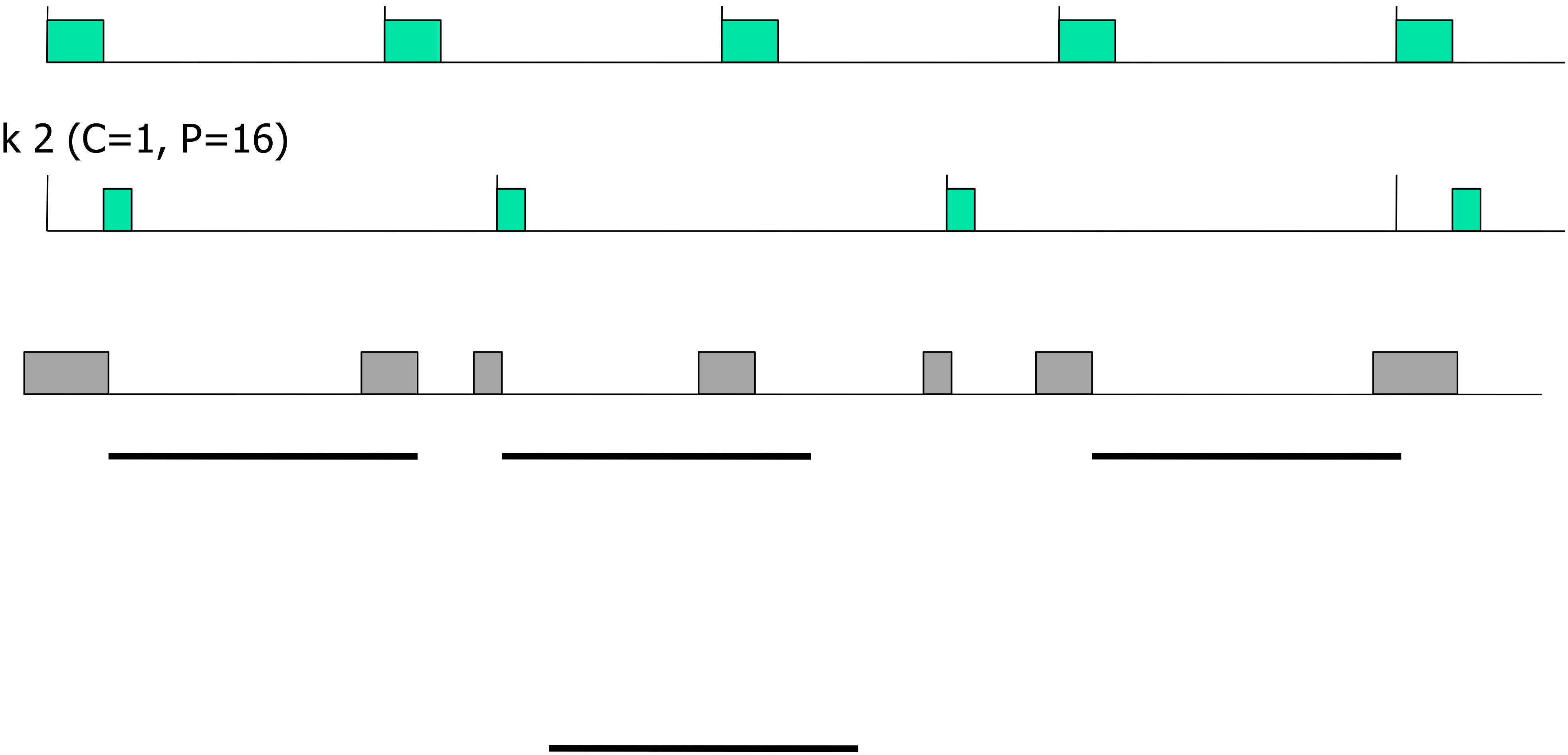
Task 2 (C=1, P=16)



The Problem with work-conserving scheduling:

Task 1 (C=2, P=12)

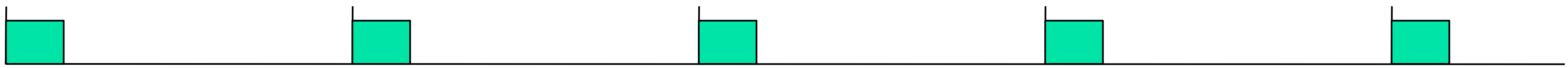
Task 2 (C=1, P=16)



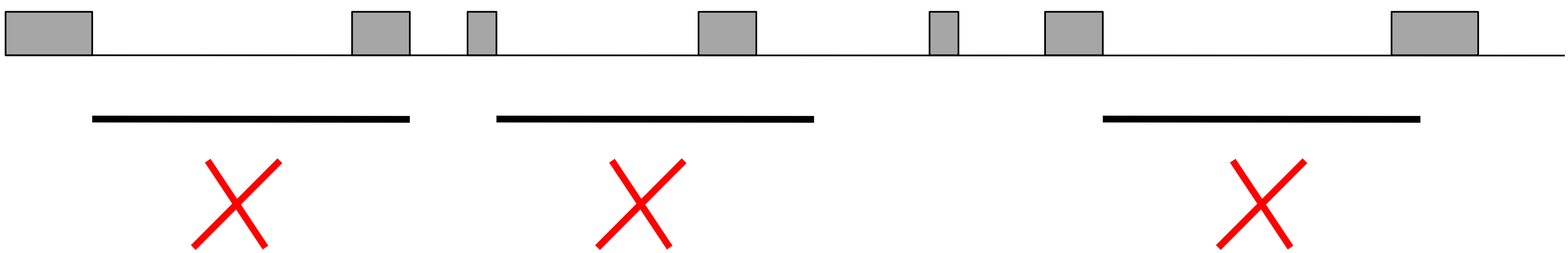
Minimum sleep period

The Problem with work-conserving scheduling:

Task 1 (C=2, P=12)



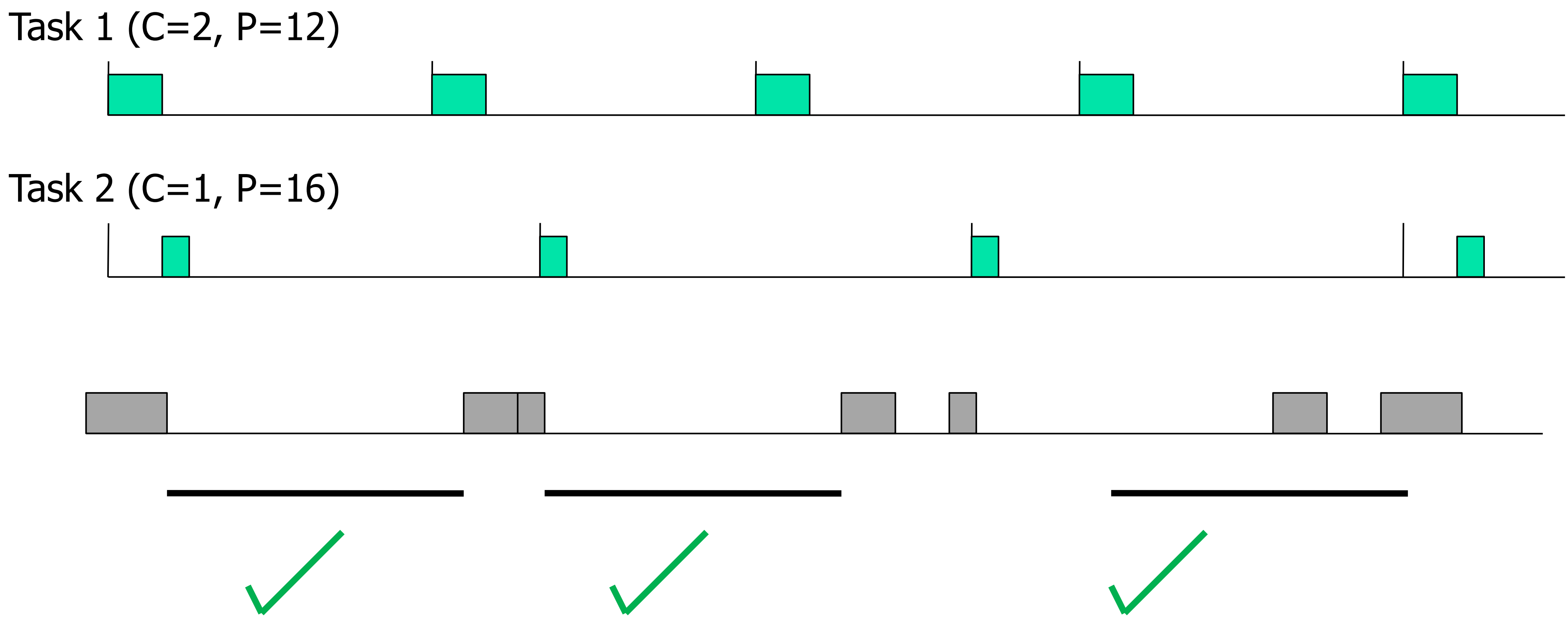
Task 2 (C=1, P=16)



No opportunity to sleep! : - (

Minimum sleep period

The Problem with work-conserving scheduling:



Solution: Must batch!

DVS on Multiple Procs

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

How many proc 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
  - $2\ \{k_f (f/2)^3 + R_0\} = k_f f^3/4 + 2\ R_0$

How many proc 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
  - $2\ \{k_f (f/2)^3 + R_0\} = k_f f^3/4 + 2\ R_0$
- The general case: n processors
  - $n\ \{k_f (f/n)^3 + R_0\} = k_f f^3/n^2 + n\ R_0$

How many proc 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?