

Speed Scaling & Sleep Management - to save energy

Prudence Wong
University of Liverpool

Bristol Algorithms Days 2010 (16 Feb)

Joint work with

Tak-Wah LAM, Hing-Fung TING (U of Hong Kong)

Lap-Kei LEE (Max-Planck-Institut Informatik)

Isaac K.K. TO (Liverpool)

Appeared in ICALP 2009

Energy Efficiency

How to save energy?



Dynamic speed (voltage) scaling

- Slow down processor whenever possible
- To run at speed s , power = s^α where $\alpha > 1$
(literature: $\alpha = 2$ or 3)

OR

Let the processor “sleeps”

Speed Scaling & Sleep Management

➤ Two states: awake or sleep

➤ Awake state

❖ power = $s^\alpha + \sigma$ when running at speed s

❖ $\sigma > 0$ is the static power

❖ speed model: infinite speed vs bounded speed

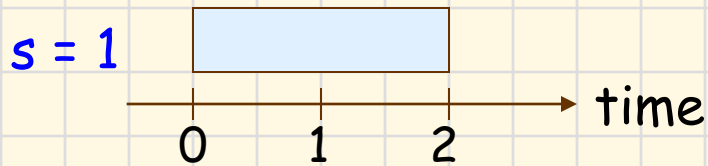
➤ Sleep state

❖ zero speed & consumes no energy

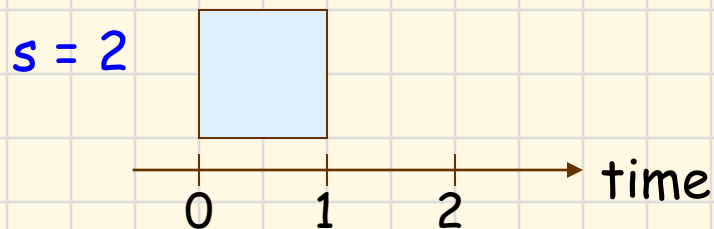
❖ wake-up to awake state requires $\omega > 0$ energy

Examples

- No sleep state ($\sigma=0$), $\alpha = 3$, a size-2 job

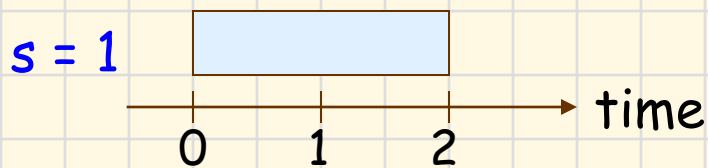


energy: $1^3 \times 2 = 2$

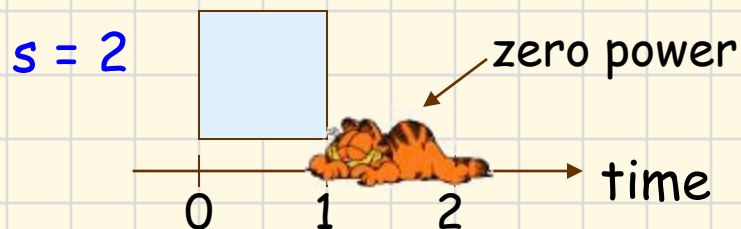


energy: $2^3 \times 1 = 8$

- With sleep state, $\sigma = 10$



energy: $(1^3+10) \times 2 = 22$



energy: $(2^3+10) \times 1 = 18$

Awake (idle/working) Or Sleep

- Further distinguish two awake states
 - ❖ **Idle:** speed = 0, power = σ
 - ❖ **Working:** speed > 0, power = $s^\alpha + \sigma$
- Job scheduling requires two components
 1. Sleep management algorithm
 - ❑ Determine **when to sleep, idle and work**
 2. Speed scaling algorithm
 - ❑ Determine **which job to run & at what speed**



Simple Ideas (that don't work)

- **idle** switch to **working** whenever a new job arrives?
 - ❖ no extra wake-up cost, sounds right?
 - ❖ adversary can release tiny jobs sporadically
⇒ never gets to sleep
- **working** switch to **sleep** whenever finishes all jobs on hand?
 - ❖ adversary can release tiny jobs right after processor sleeps ⇒ keeps paying wake-up cost

Orthogonal Objectives

- Energy efficiency: run jobs slower + sleep
- Quality of service: run jobs faster + awake
- Flow time: time elapsed from release to complete
- We want: small flow time, small energy
- Objective: minimize $F+E$ [Albers & Fujiwara STACS'06]
 - ❖ users are willing to pay one unit of energy to reduce ρ units of flow time

Online Scheduling

- Jobs arriving at unpredictable time
 - ❖ Job information known on arrival, no future information
- Preemption is allowed
- Competitive ratio
 - ❖ An online algorithm A is **c-competitive** if for any job sequence I ,

$$A(I) \leq c \text{OPT}(I)$$

where $A(I)$ and $\text{OPT}(I)$ are F+E of A and OPT on I

Results

➤ Without sleep state:

- ❖ BPS [Bansal, Pruhs, Stein SODA'07]
- ❖ AJC (active job count) [Lam, Lee, To, W. ESA'08], [Bansal, Chan, Pruhs SODA'09]

➤ With sleep state:

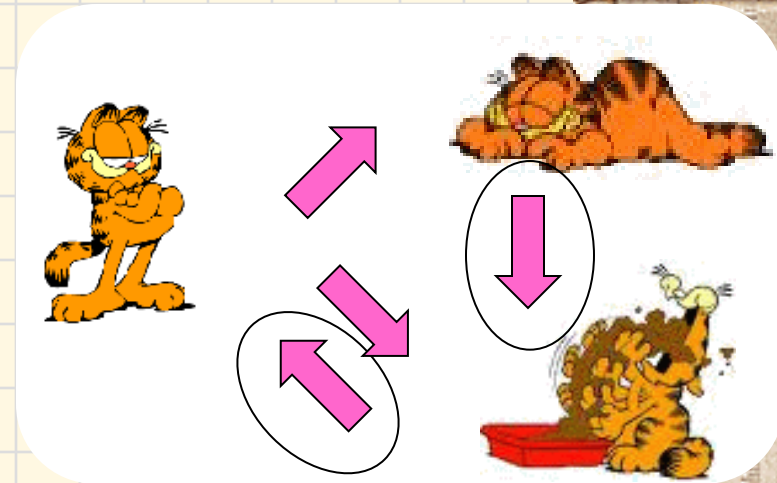
- ❖ sleep management algorithm IdleLonger & adapted AJC [Lam, Lee, Ting, To, W. ICALP'09]

	competitive ratio	
w/o sleep state	$O\left(\frac{\alpha}{\log \alpha}\right)$	3
with sleep state	$O\left(\frac{\alpha}{\log \alpha}\right)$	

IdleLonger

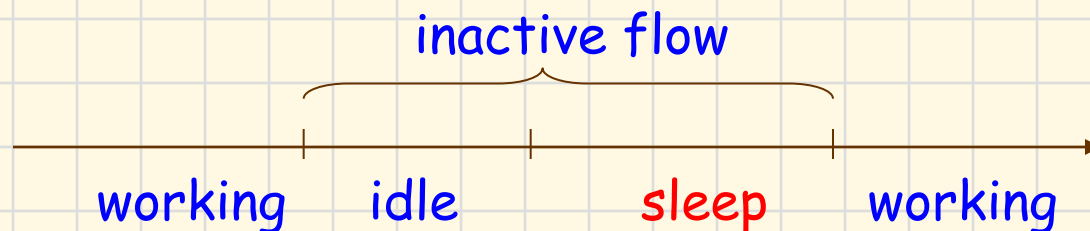
➤ From working to idle

- ❖ continue working when there is unfinished job
- ❖ when all jobs completed, switch to idle



➤ From sleep to working

- ❖ wait for enough jobs, then wake-up
- ❖ **inactive flow**: flow accumulated by new jobs
- ❖ switch to working if **inactive flow = wake-up energy**



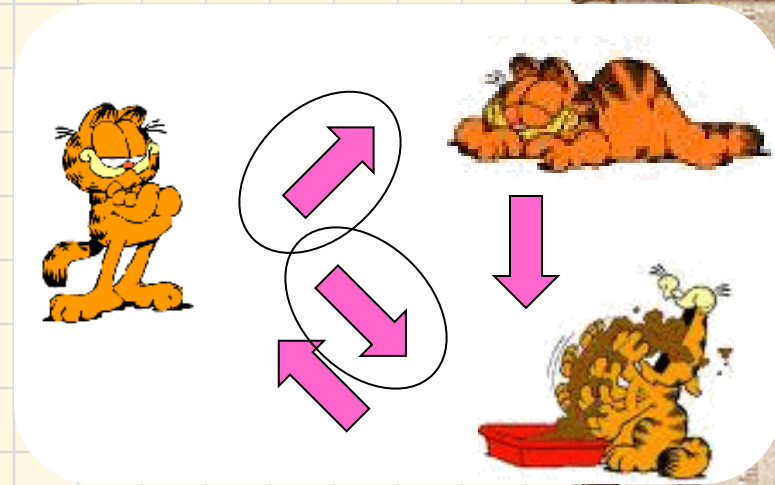
IdleLonger

➤ From idle to working/sleep?

- ❖ don't switch to working as soon as a new job arrives
- ❖ don't idle too long before switching to sleep

➤ Main idea: **sleep with the presence of jobs**

- ❖ looks counter-intuitive
- ❖ but gives a good balance of energy & flow time



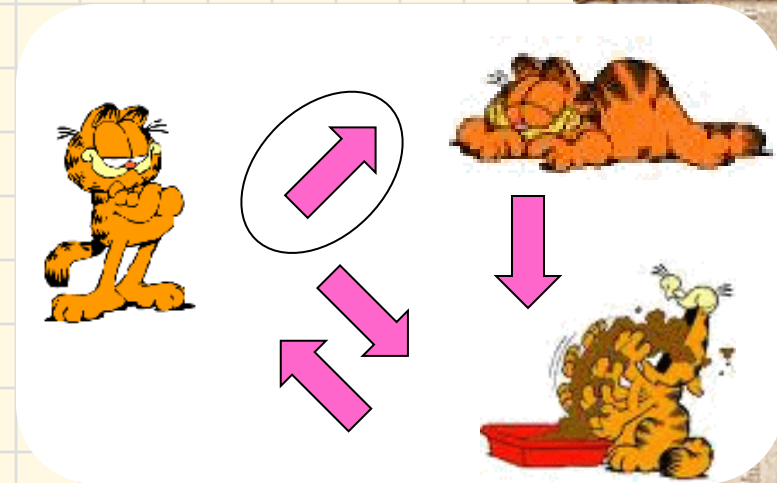
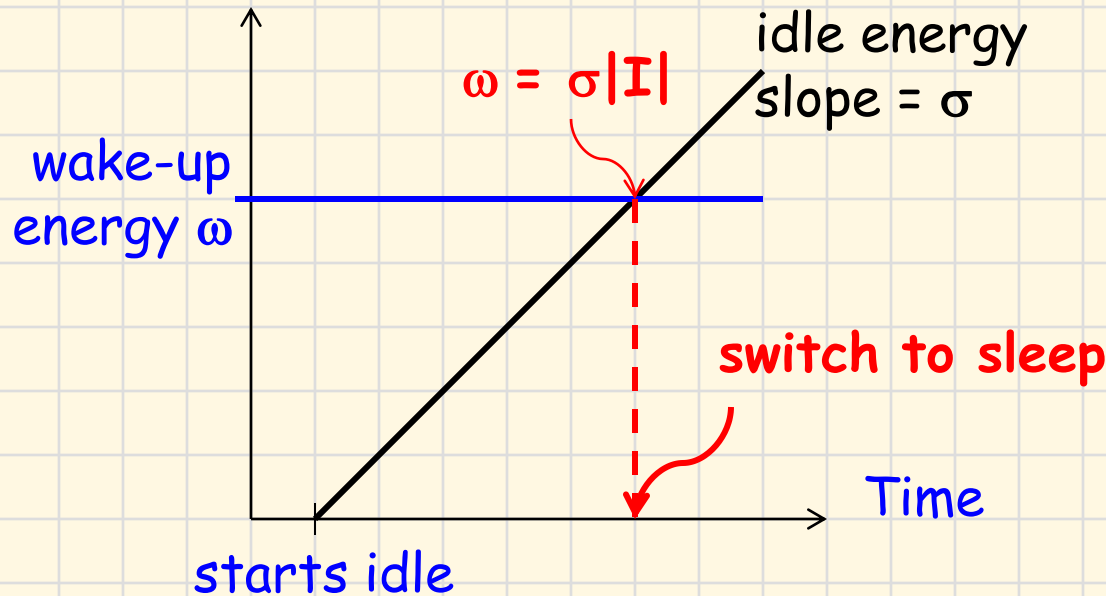
IdleLonger

➤ From idle to working/sleep?

➤ **From idle to sleep**

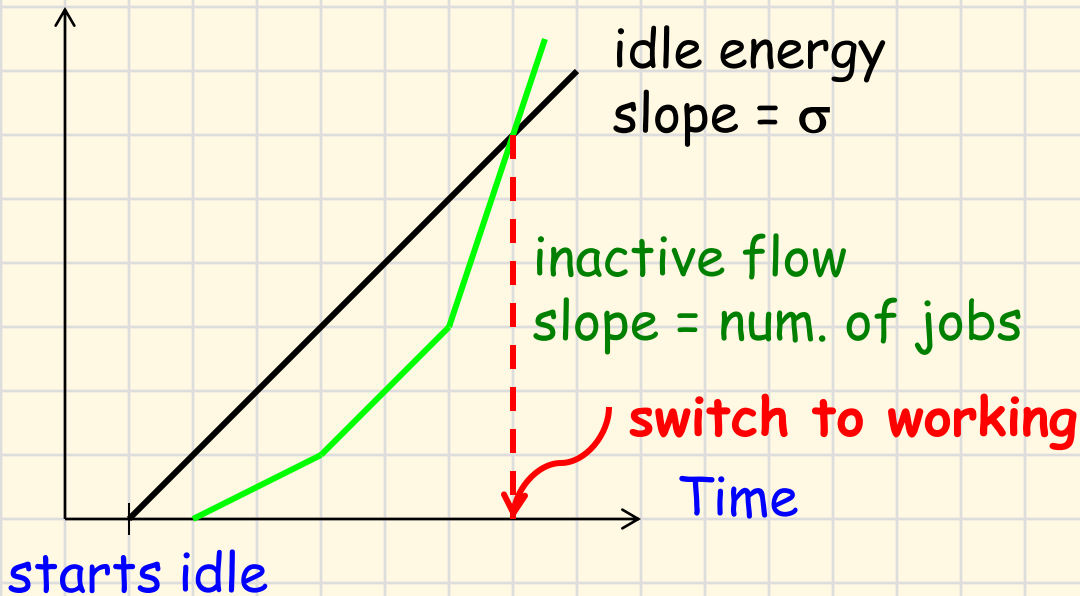
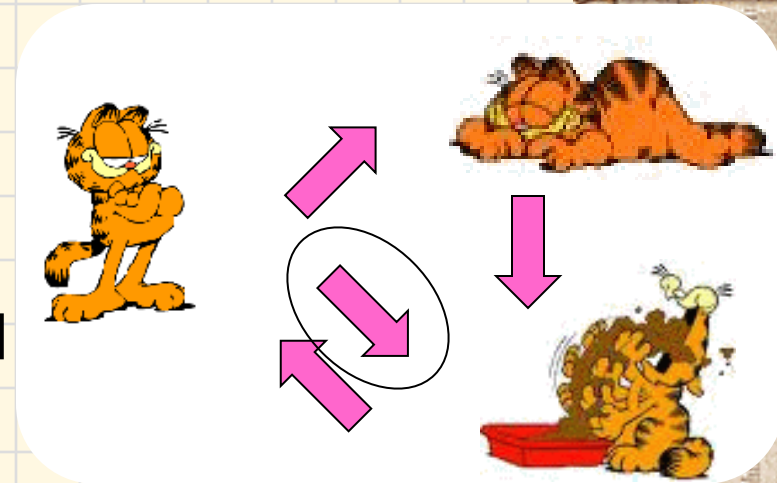
❖ **idle energy:** grows linearly with the idle period length

❖ switch to sleep if
idle energy = wake-up energy (cf. ski-rental problem)



IdleLonger

- From idle to working/sleep?
- **From idle to working**
 - ❖ **inactive flow:** flow accumulated by new jobs
 - ❖ switch to working if inactive flow = idle energy



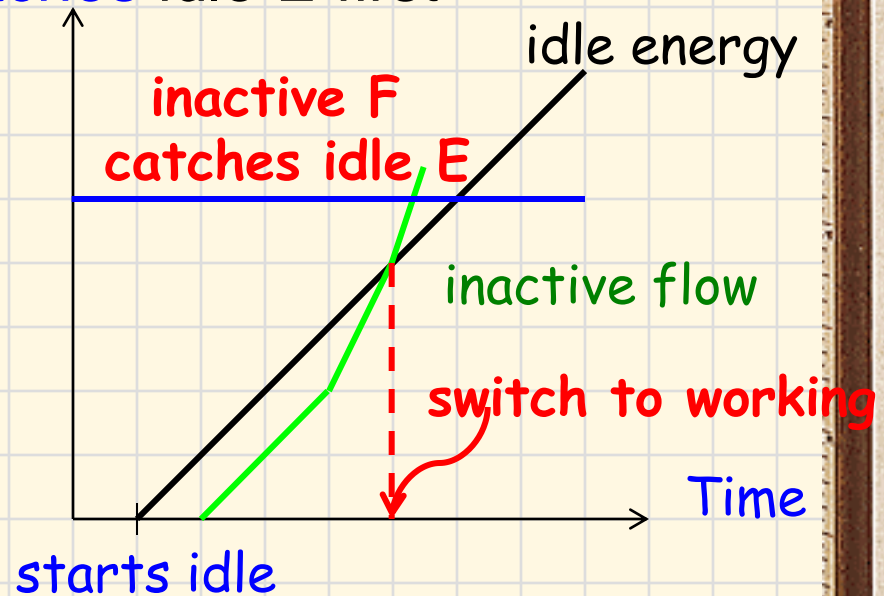
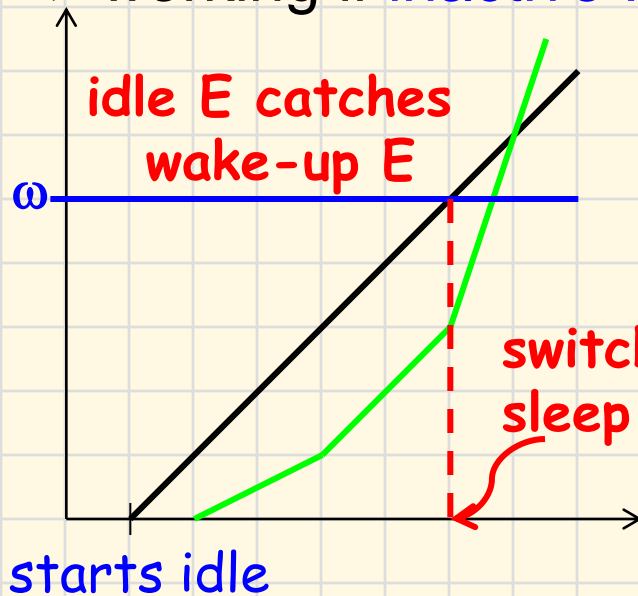
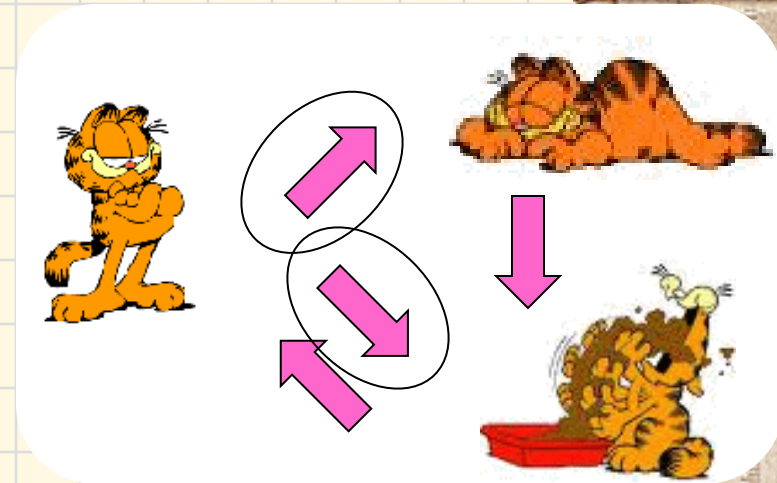
IdleLonger

➤ From idle to working/sleep?

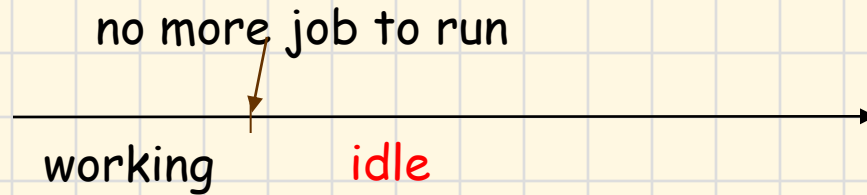
- ❖ idle E chasing wake-up E
- ❖ inactive F chasing idle E

➤ How to decide?

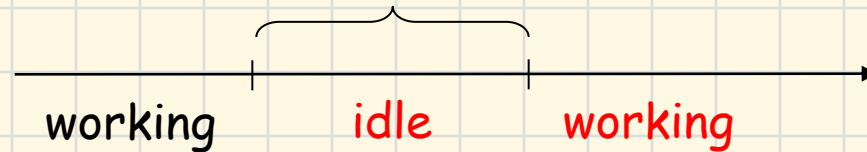
- ❖ sleep if idle E catches wake-up E first
- ❖ working if inactive F catches idle E first



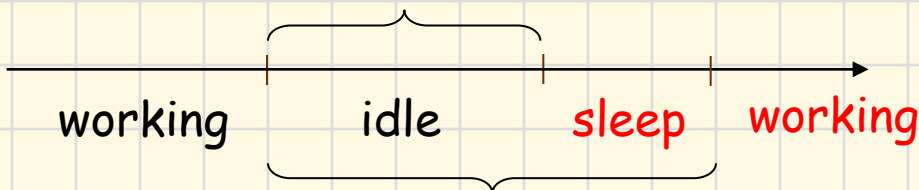
IdleLonger



inactive flow = idle energy < wake-up energy



inactive flow < idle energy = wake-up energy



inactive flow = wake-up energy

Adapting AJC

➤ AJC (w/o sleep state)

- ❖ which job? job with smallest **remaining work**
- ❖ what speed? $n^{1/\alpha}$, where n = num of unfinished jobs
 - at any time, flow & energy incurred at same rate
- ❖ bounded speed? cap at max speed T

➤ Sleep-aware AJC (with sleep state)

- ❖ working also requires static power σ
 - balance cannot be maintained using AJC
 - too slow if σ is large
- ❖ new speed: $(n+\sigma)^{1/\alpha}$

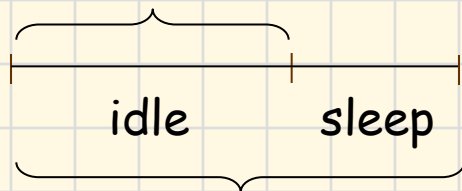
Framework of Analysis

- isolate the analysis of sleep management & speed scaling
- divide F+E into **inactive cost** & **working cost**
- IdleLonger – using any speed scaling
 - ❖ **inactive cost:** incurred during idle & sleep
 - flow time + stay-awake energy + wake-up energy
 - ❖ O(total cost of OPT)
- adapted-AJC – using any sleep management
 - ❖ **working cost:** F+E incurred during working
 - ❖ O(total cost of OPT) + O(inactive flow)

Inactive Cost

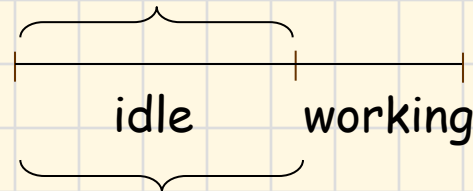
- inactive flow, idle energy, wake-up energy
 - ❖ **W**: wake-up energy
 - ❖ **E_{IW}** : idle energy of idle interval switching to working
- inactive cost $\leq 3W + 2E_{IW}$

idle energy = wake-up energy



inactive flow = wake-up energy

idle energy



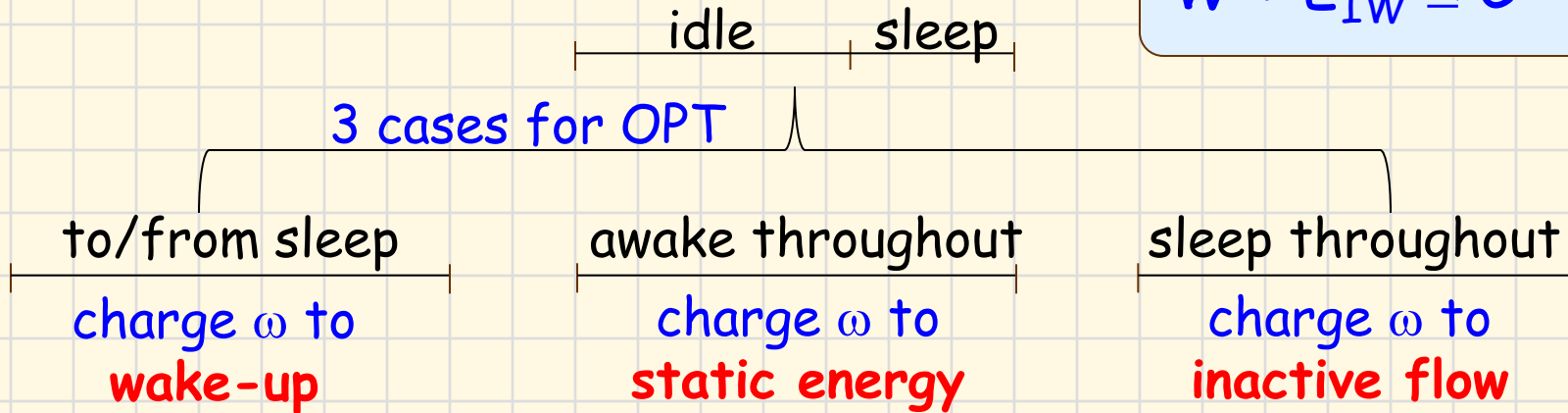
inactive flow = idle energy

Inactive Cost – counting argument

➤ wake-up energy W

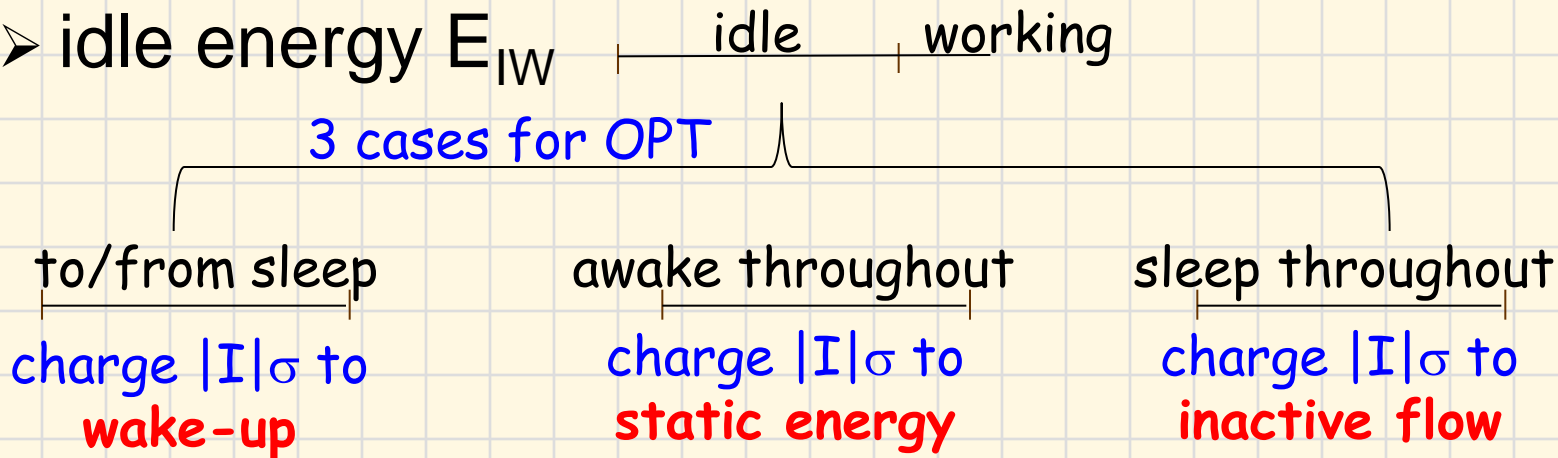
$$W + E_{IW} \leq C^* + 2W^*$$

3 cases for OPT



➤ idle energy E_{IW}

3 cases for OPT



Working Cost

- adapt analysis of AJC
- potential & amortization analysis



Bounded Speed Model

- Once overslept, cannot catch up the delay by speeding up arbitrarily
- IdleLonger needs to wake up earlier
 - ❖ when the number of jobs exceeds static power σ
- For speed scaling, capping speed at the maximum speed is enough

Multiple Sleep States

➤ m sleep states

➤ In **sleep-i** state

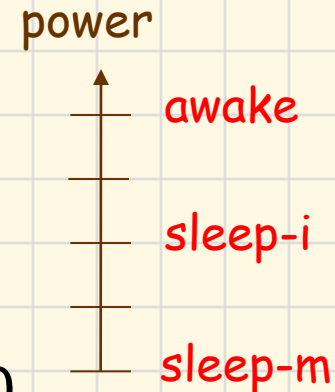
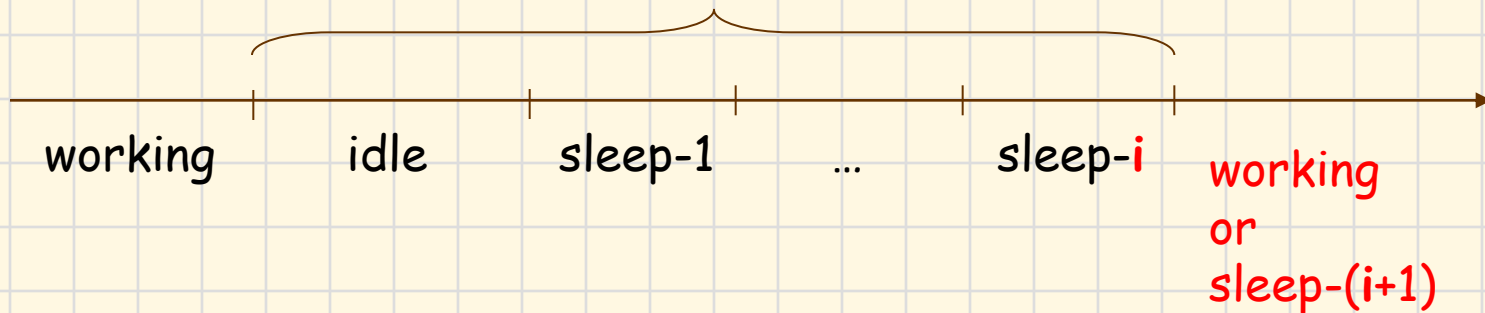
❖ zero speed

❖ static power σ_i : $\sigma > \sigma_1 > \sigma_2 > \dots > \sigma_m = 0$

❖ wake-up energy ω_i : $0 < \omega_1 < \omega_2 < \dots < \omega_m$

➤ IdleLonger: generalize idea of idle state

inactive flow vs idle energy



Remarks

➤ Related problems

- ❖ Non-clarivoyant scheduling
 - job size is not known until it is finished
- ❖ Deadline scheduling
 - concern is maximizing throughput and minimizing energy
- ❖ Multi-processors
- ❖ Switching overhead

Thank You!

