Speed Scaling in the Non-clairvoyant Model

SPAA'15 (Best Paper Award)

Energy efficiency

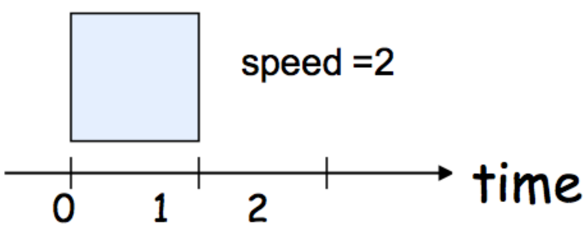


- For design of mobile devices, energy efficiency is a major concern.
- How to save energy? Dynamic speed (voltage) scaling
  - Slow down processor whenever possible.

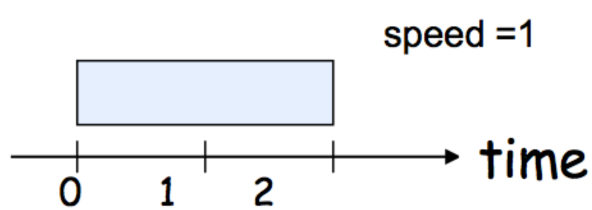
Model

- At any time, processor speed **s** is adjustable.
- power = $\mathbf{s}^{\alpha}$, where $\alpha > 1$ ($\alpha \sim 3$).

E.g., $\alpha = 3$ & run a job with 2 units of work.



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



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

Online job scheduling

- Schedule a set of jobs on a processor.
- Jobs arrive online, i.e., no future information.

Quality of Service: Total flow time of jobs

- Flow time of J = completion time c(J) - release time r(J)
  - measure how long to wait before a job completes
- Jobs may have weights to indicate their importance.
  - QoS measure: total weighted flow time.

Minimize flow time plus energy

- We want
  - small total flow time **F** : run job faster
  - small energy consumption **E** : run job slower

- Combined objective **F+E** [Albers and Fujiwara; STACS06]
  - From economic viewpoint, users are willing to pay one unit of energy to reduce ρ units of flow time
  - Minimize $\mathbf{F} + \rho\, \mathbf{E}$
  - WLOG, we can assume $\rho=1$

Algorithm & Performance

Scheduling algorithm needs to decide at any time:

1. which job to run (job selection)
2. at what speed the job is run (speed scaling)

Performance of online algorithm ALG:

Competitive analysis

- Compare cost of ALG with optimal offline algorithm **OPT**.
  - offline: **OPT** has complete information in advance.
- ALG is c-competitive if for any job sequence I,

$$\mathbf{ALG}(I) \leq c\ \mathbf{OPT}(I)$$

Clairvoyant & Non-clairvoyant settings

Clairvoyant setting:

- When a job arrives, the job size is known.
  - E.g., web-server serving static documents

Non-clairvoyant setting:

- When a job arrives, job size is not known until it completes.
  - E.g., operating systems

- Non-clairvoyant algorithm is applicable to the clairvoyant setting.

Notations

Input:

- Job $j \in J$
- release time: $r[j]$
- volume: $V[j]$
- density: $\rho[j]$
- weight of job j: $\rho[j] \times V[j]$

Setting:

- speed: s
- power function: $P = s^\alpha$

Algorithms (single job)

Clairvoyant algorithm (ALG_C):

- Instantaneous Power = Remaining Weight

Non-clairvoyant algorithm (ALG_NC):

- Instantaneous Power = Processed Weight by ALG_C

Both algorithms determine jobs according to first-in first-out (FIFO).

Objective

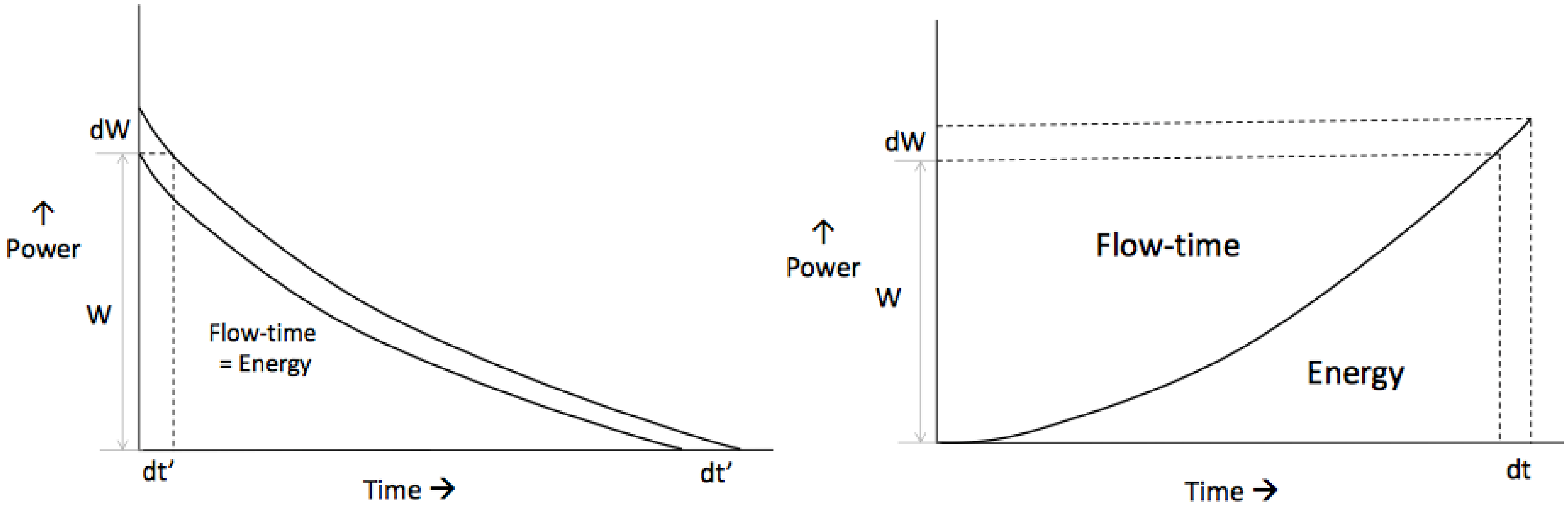
Minimize: Energy + Flow time

- Energy = integral of power curve (time-power)
- (fractional) flow time = $\rho[j] \cdot \int_{t=r[j]}^{\infty} \overline{V}(t)[j]\, dt$

for <u>single job</u>:

ALG_C: Instantaneous Power = Remaining Weight

ALG_NC: Instantaneous Power = Processed Weight by ALG_C



(a) The clairvoyant power curve (b) The non-clairvoyant power curve

Uniform Density (multiple jobs)

Clairvoyant algorithm (ALG_C):

- Instantaneous Power = Remaining Weight

Non-clairvoyant algorithm (ALG_NC):

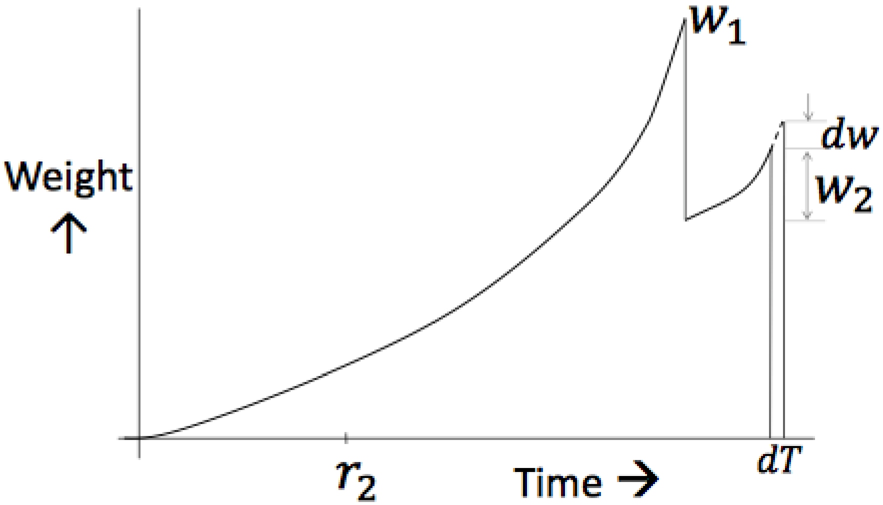
- Instantaneous Power = Remaining Weight by ALG_C + Processed Weight by ALG_NC (at time t)

Both algorithms determine jobs according to first-in first-out (FIFO).

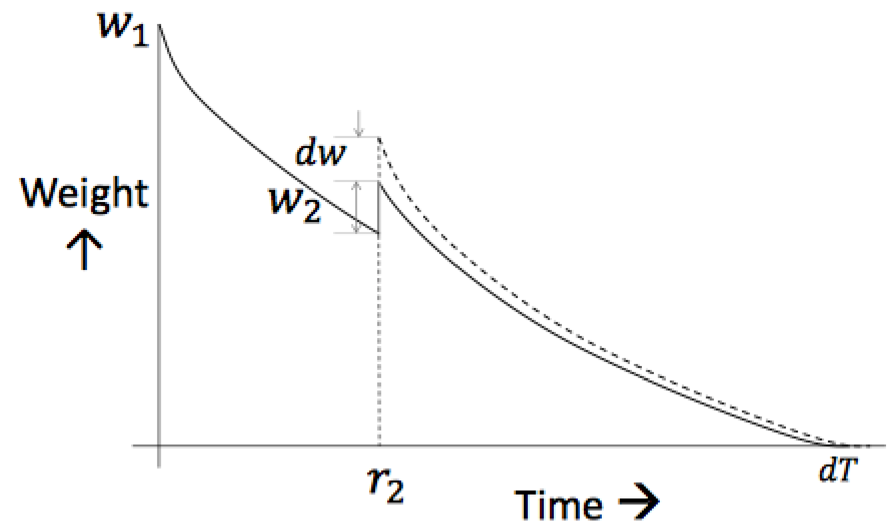
Example

Lemma 1: ALG_NC and ALG_C have the <u>same</u> energy consumption

Lemma 2: ALG_NC has at most $1/(\alpha - 1)$ times <u>flow time</u> of ALG_C's <u>flow time</u>



(a) The change in the non-clairvoyant algorithm upon processing an extra dw weight of job 2 which takes an extra time of dT. Job 2 is released at r_2 and has weight w_2 currently. Job 1 is released at time 0 and has weight w_1, all of which has been processed.



(b) The change in the run of the clairvoyant algorithm due to an extra dw weight of job 2. Here the speed of the algorithm changes all the way from time r_2 to the end. The extra time taken dT is however the same as in the non-clairvoyant case.