A Tentative Proposal for Energy Complexity of Algorithms

M. Chrobak
U. California @ Riverside

Some Motivations

1. If we want to study energy complexity of algorithms, algorithms must be able to control energy usage

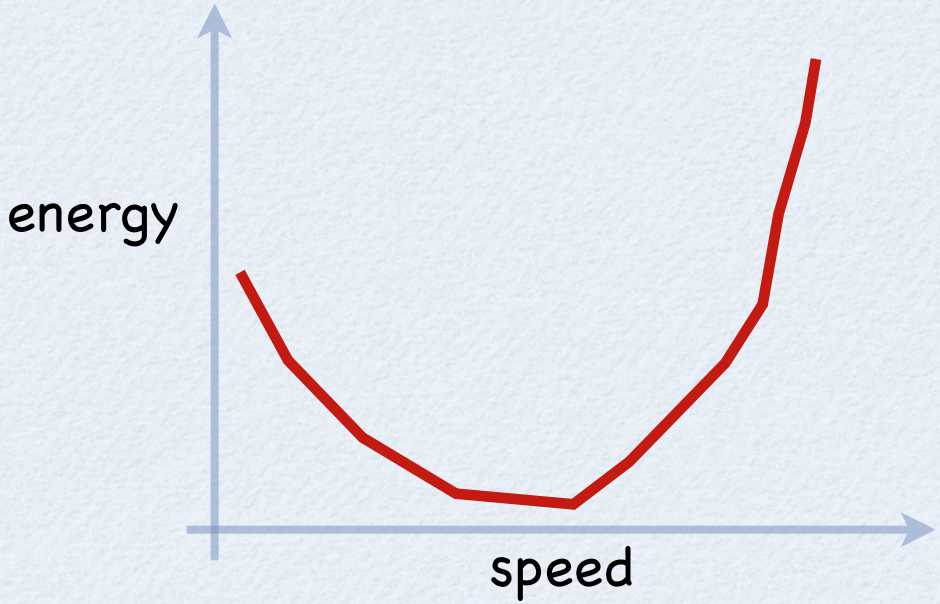
2. Most work in theory focuses on workload scheduling: speed scaling and power-down, but....

3. If we allow an algorithm to control speed, it will run at speed -> 0 with energy -> 0, which is

- not realistic
- not interesting

Some Motivations

4. Typical energy usage as function of speed:



5. Other system components need energy too:
   - memory (~ 20% - 30%). We must pay somehow for data storage
   - other (constant overhead, cooling, yaddi-yadda...)

Power Usage

Processor: power = CV^2f where

- C = capacitance
- V = voltage
- f = frequency

where V ~ f so power = s^{α}, s = speed, $\alpha \sim 3$.

Memory: messy ....

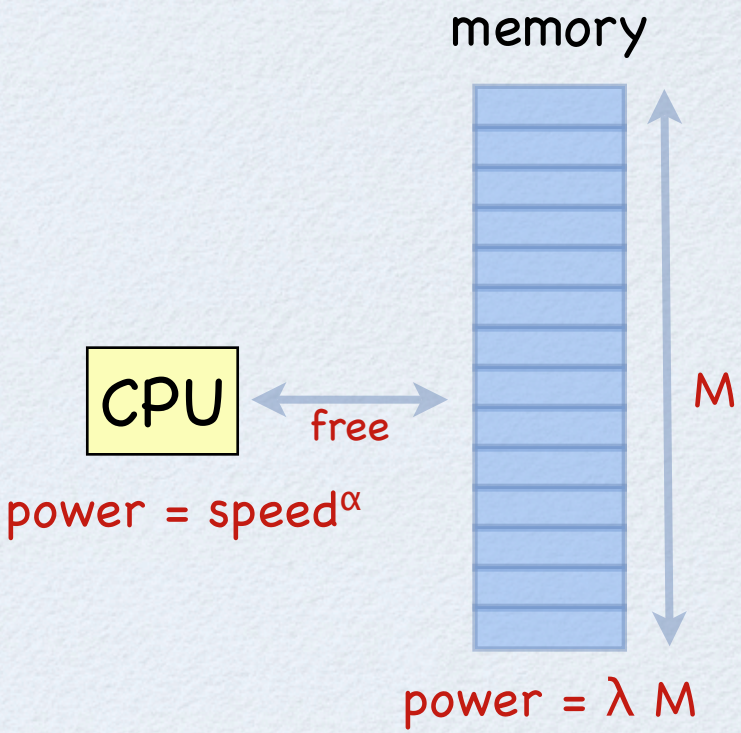
- different types of RAM, caches, disk, you-name-it
- different states
- storage (refresh energy), movement ...

So assume power = λ M , M = active memory

Model: ARAM = Adjustable RAM

Assumptions

- algorithm can adjust speed (unlimited, just for fun, but you can also assume bounded speed)
- algorithm can turn on/off individual memory locations (for free, instantaneously)
- data initially in memory (how to get rid of this ???)
- who knows what other assumptions ...



Example 1: Sorting

running at speed s:

energy per step = cpu-energy + memory-energy

$= s^{\alpha}/s + \lambda n/s$

after minimizing

$s = (\lambda n/(\alpha-1))^{1/\alpha}$

so

$E = s^{\alpha}(n\log n/s) = A_{\alpha}\ \lambda^{1-1/\alpha} \cdot n^{2-1/\alpha}$

$T = n\log n/s = B_{\alpha}\ \lambda^{-1/\alpha} \cdot n^{1-1/\alpha} \log n$

For $\alpha = 3$ and constant λ

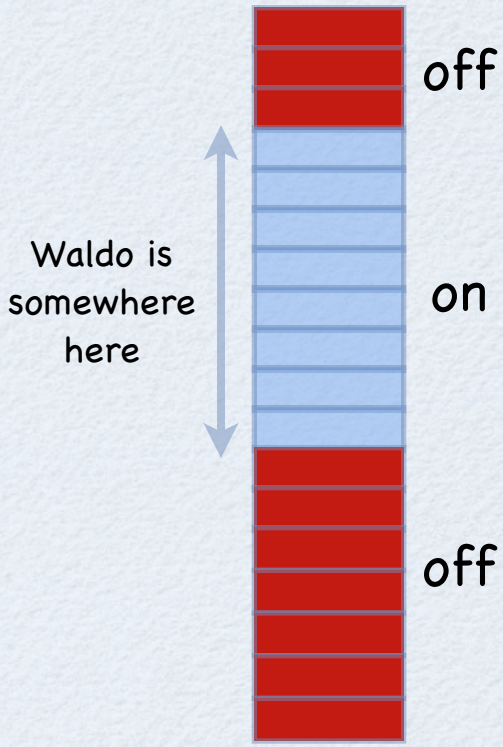
$E \sim n^{5/3}$

$T \sim n^{2/3} \log n$

Example 2: Binary Search

Regular binary search, plus:

- use speed s_i at step i
- turn off memory where Waldo cannot be located



Example 2: Binary Search

energy in step i = $(s_i^{\alpha} + \lambda n/2^i)/s_i$

minimizing, we get

$$s_i = (\lambda n/(\alpha-1)2^i)^{1/\alpha}$$

total energy $E = \Sigma_i(\lambda n/(\alpha-1)2^i)^{(\alpha-1)/\alpha}$

$$= A_\alpha \lambda^{(\alpha-1)/\alpha} \cdot n^{(\alpha-1)/\alpha}$$

time $T = \Sigma_i(\lambda n/2^i)^{-1/\alpha}$

$$= B_\alpha \lambda^{-1/\alpha}$$

For $\alpha = 3$

$$E \sim n^{2/3}$$

$$T \sim 1$$

What's Next?

1. Does anything interesting happen if the algorithm needs a LOT of memory? Will re-computation be useful?

2. Max speed = 1??

3. How to get rid of the assumption that input is in the memory?

4. Do some reading ...

More Musings About Memory: Online Minimization of Energy Consumption

M. Chrobak
U. California @ Riverside

Problem 1: Online Memory Consolidation

States: active, standby, nap, power-down, off

with different power usage and transition costs (energy, delay)

All states except "off" retain values, but memory can be accessed only in "active" state

If you can control each memory location:

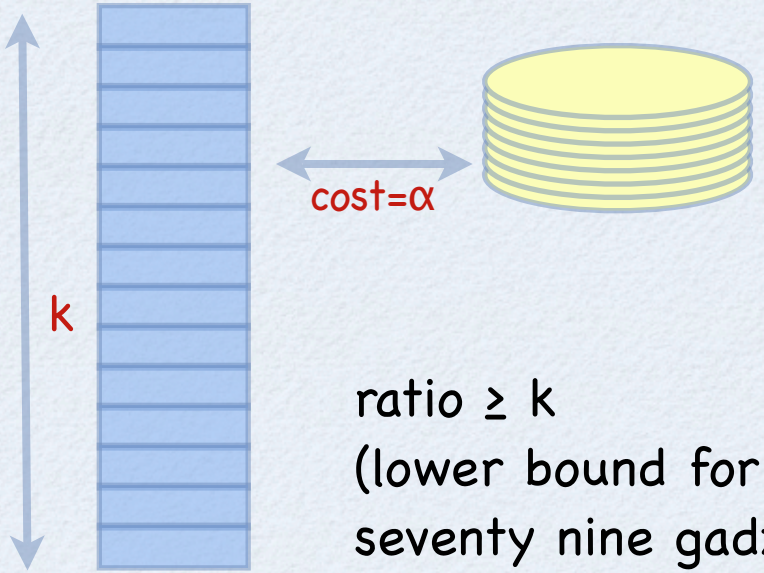
problem = multi-state ski-rental (studied)

But you can't. Suppose you have K blocks, M locations each. You can consolidate: move values between blocks (at extra cost).

Formalize, and what's the competitive ratio?

Problem 2: Online Disk Energy Usage Minimization

RAM: cost = 0 (or itsy bitsy)



Disk: cost = β/step to spin
B to wakeup

ratio ≥ k
(lower bound for paging, plus repeat each request seventy nine gadzillion times)

Extend to a network of computers ("cooperative caching") ??

"A lightweight approach to reducing energy management delays in disks", G. Wang et al., this conference