Introduction

Chapter 1 Prologo
Lecture Notes
prof. Paolo Ferragina

Algorithm Engineering

- Teachers

Paolo Ferragina www.di.unipi.it/~ferragin
Linda Pagli www.di.unipi./~pagli
office number 277
office hours by appointment (email)

Department of Computer Science.
Website of the course:
didawiki.di.unipi.it/doku/php/
magistraleinformaticanetworking/ae/start

Preliminaries

- In order to be able to understand these lectures it is necessary to know basics in algorithms and computational complexity.
- Know how to evaluate algorithms in the RAM model.
- Know how to write a program.

Otherwise study the book:

Cormen Leiserson Rivest Stein,: Introduction to Algorithms. Worldwide famous book used in the most prestigious universities.

Lecture Notes

- Almost all topics considered are contained into the Lecture Notes.
- Otherwise references will be given.
- Treated topics can be sometimes hard hence you will need a personal effort to understand them. Following the lectures is not enough.
- We will use a pseudo-code (similar to java or C) to define algorithms to be able to discard details but… you are entitled to know them!!!

Interesting problems

- We will not use a formal approach.
- We will analyze solutions for some interesting problems arising from real/useful applications.
- We will study solution of improved efficiency and increasing sophistication.
- Before: Model of Von Neumann RAM model
- Last 10 years: 2 main changes.
  - The architecture of modern PC are more and more complex
  - Explosion of input size

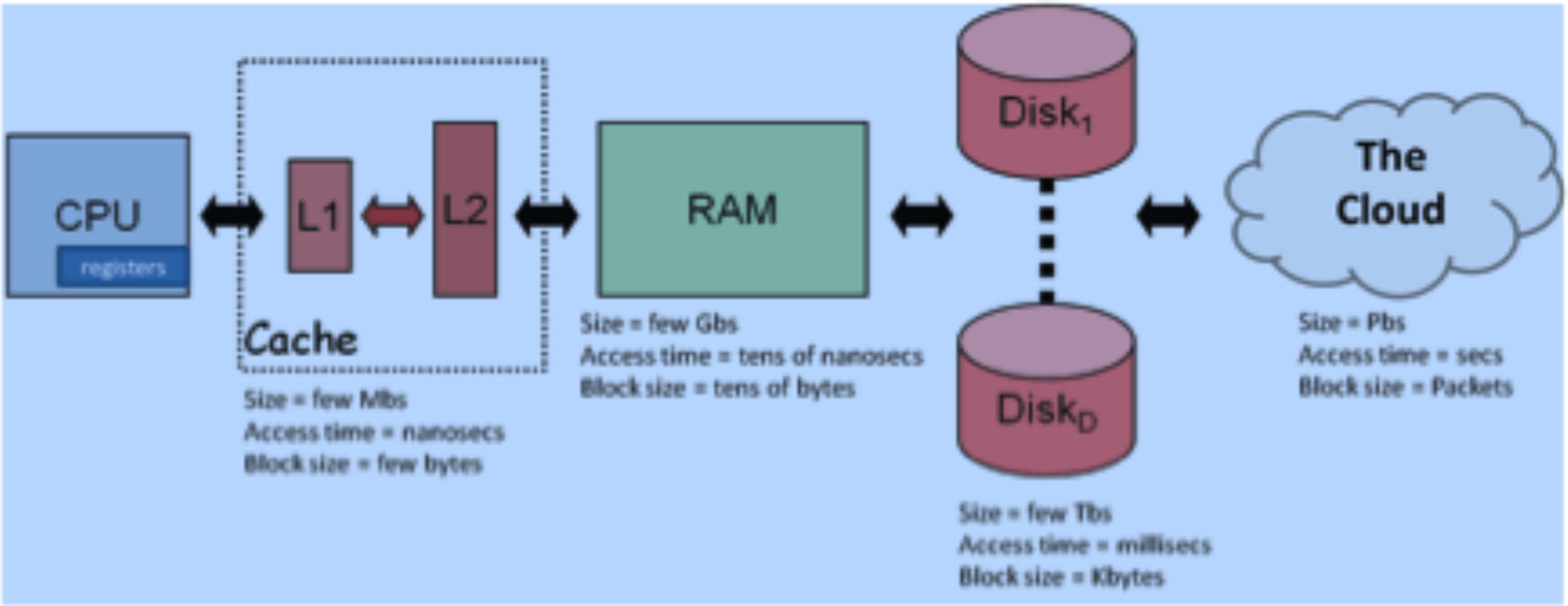
The RAM model is more and more unsatisfactory!

Algorithm Engineering

- Derive efficient algorithms for the new models.
- Adapt old efficient algorithms to the new models
- Exploit techniques coming from other memory models such as parallel computation.
- Derive general techniques to be adopted in different situations.

Example: Compute the sum of the integers stored in the Array A[1, n] working in a modern PC.

A modern PC



Size = few Mbs
Access time = nanosecs
Block size = few bytes

Size = few Gbs
Access time = tens of nanosecs
Block size = tens of bytes

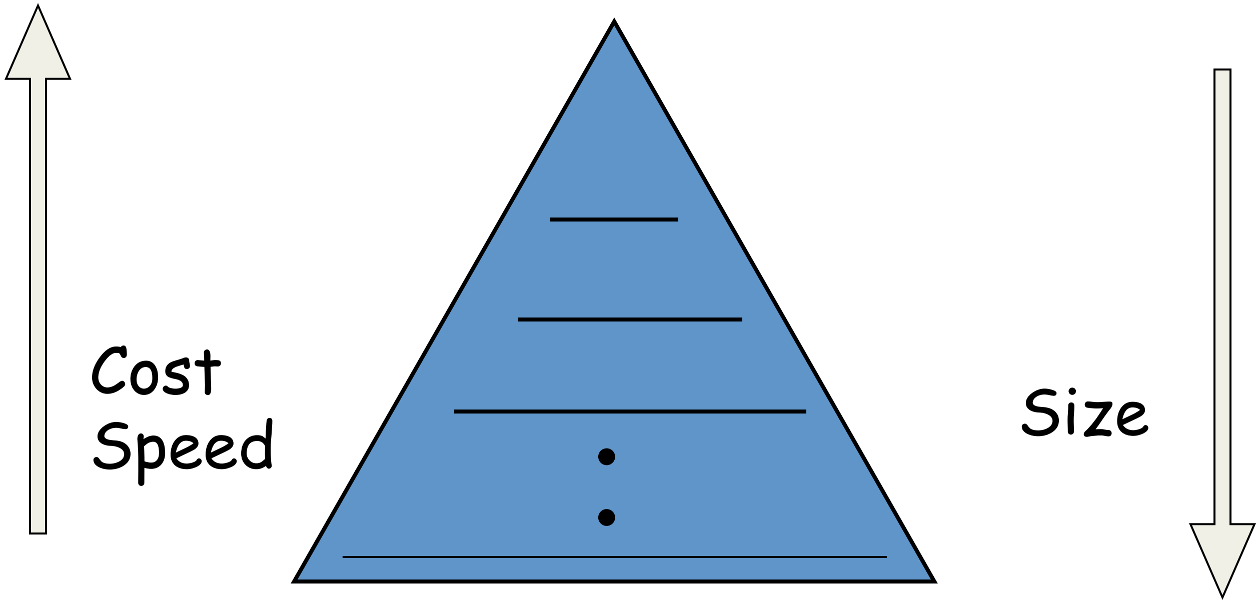
Size = few Tbs
Access time = millisecs
Block size = Kbytes

Size = Pbs
Access time = secs
Block size = Packets

.

PAST: Main memory VS External memory

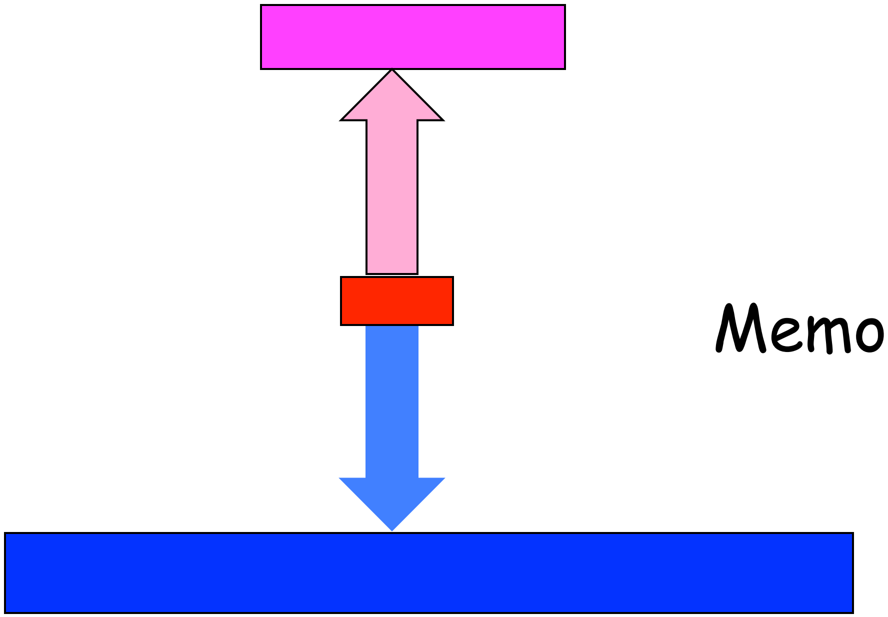
NOW: Memory hierarchy



Cost
Speed

Size

The access mechanisms are the same



Memory block

Two adjacent levels are considered.
Data are transferred in blocks of fixed size,
called PAGES

Locality principle

- The block transfer is an expensive operation.
- Apply the locality principles to organize efficient accesses to blocks

Temporal locality: It is probable that an already requested object will be requested again in the future.

Spatial locality: It is probable that objects close to already referred ones will be requested again in the future.

2-level memory model (disk model)

- B= block (page) size
- M= internal memory size

How to evaluate the complexity of an algorithm?

number of I/Os operations

- Our algorithm takes n/B I/Os operations is optimal.
- It is independent from the block size. Very important feature for an algorithm. Cache-oblivious.

Modern PC architecture

- Nanoseconds suffice to access the caches
- Milliseconds to access data from disks.
- I/O bottelneck
- Engineering: try to reduce the impact of I/O bottleneck handling large datasets.
- Good algorithms design surpass the best technology advancements !

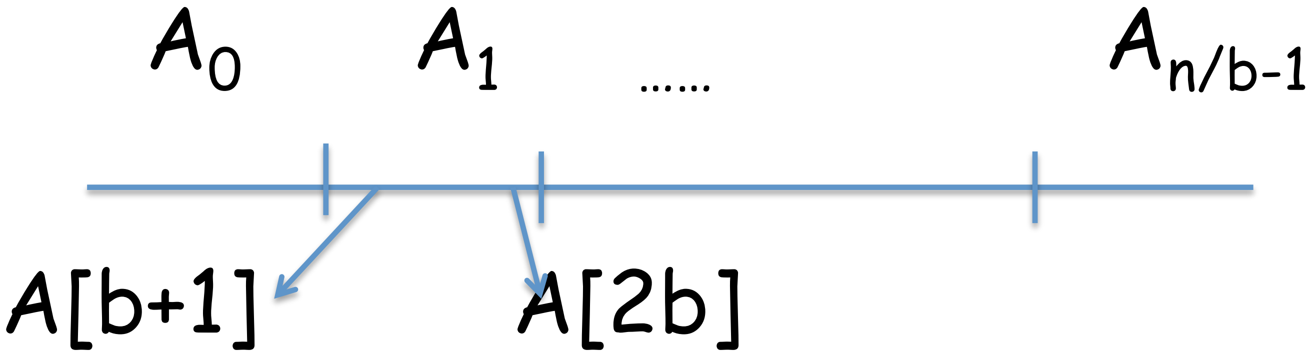
Analyze complexity in modern PC: Example

- Compute the sum of integers stored in array $A[1, n]$.
- Scan and accumulate in a variable→ $\Theta(n)$
- Define a family of algorithms $A_{s,b}$

  the patterns to access the items are different according to s and b.
- A is divided into blocks A_j $0 \leq j \leq n/b-1$ of size b (b items)

Analyze complexity in modern PC: Example

- $A_j = A[j*b+1, (j+1)*b]$, $0 \le j \le n/b$



- Sum all items of a block before moving to next block s that is s blocks apart to the right. .
- A is considered cyclic.

Analyze complexity in modern PC: Example

- b block size; s number of blocks of the jump.
- s must be co-prime with n/b in order all blocks are considered.
- Otherwise: n/b=9, s=3 the same 3 blocks are examined cyclically.
- If s is co-prime with n/b: [s×i mod n/b] generates a permutation of [0, 1, …, n/b-1] hence all blocks in *A* are touched.
- Varying s and b we can sum according to different patterns of memory accesses.

Analyze complexity in modern PC: Example

- Sequential scan s=1, b=1.
- Block-wise scan b=B.
- And/or random-wise access large values of s.
- All algorithms in $A_{s,b}$are equivalent: they read and sum exactly n integers.
- Complexity:
- s=1: $A_{1,b}$ scan A rightwards and independently from the b value takes O(n/B) I/O's
- As s and b changes situation complicates!

Analyze complexity in modern PC: Example

- s=2, b<B, b divides B. Every block B consists of B/b smaller, logical blocks of size b.
- $A_{2,b}$ examines only half of them because s=2.

  Page is half utilized! 2n/B I/O's.
- For any s : sn/B I/O's.

The formula is an approximation to the real case: all I/O's are considered equal while in reality the cost changes from sequential and random I/O's.

But the sufficiently good and widely adopted in literature. The 2-level memory model will be adopted almost in all cases.

Algorithm Engineering

How to turn

theoretically efficient algorithms

into

practically efficient code!