I/O-Efficient Algorithms and Data Structures

Spring 2008

Rolf Fagerberg

Course

Lectures:

- Theoretical (DM02/DM507++).
- New stuff: 1995-2007.
- Aim: General principles and methods.

Project work:

- Several small/medium programming projects (3 ECTS in total).
- Aim: Hands-on.

Course

Literature:

- Based on lecture notes and articles.

Prerequisites:

- DM02/DM507 Algorithms and Data Structures.

Duration:

- 3rd and 4th quarter.

Credits:

- 10 ECTS (including project).

Exam:

- The projects (pass/fail), oral exam (7-step scale).

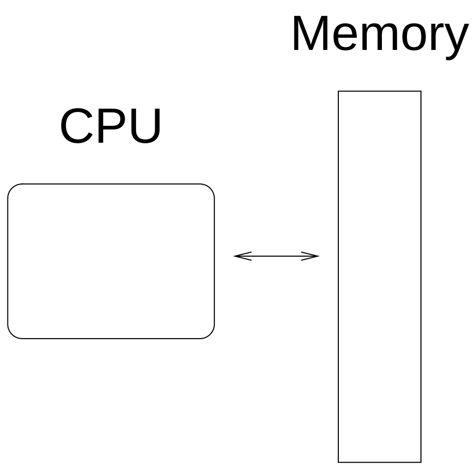
Statement of Aims

After the course, the participant is expected to be able to:

- Describe general methods and results relevant for developing I/O-efficient algorithms and data structures, as covered in the course.
- Give proofs of correctness and complexity of algorithms and data structures covered in the course.
- Formulate the above in precise language and notation.
- Implement algorithms and data structures from the course.
- Do experiments on these implementations and reflect on the results achieved.
- Describe the implementation and experimental work done in clear and precise language, and in a structured fashion.

Analysis of algorithms

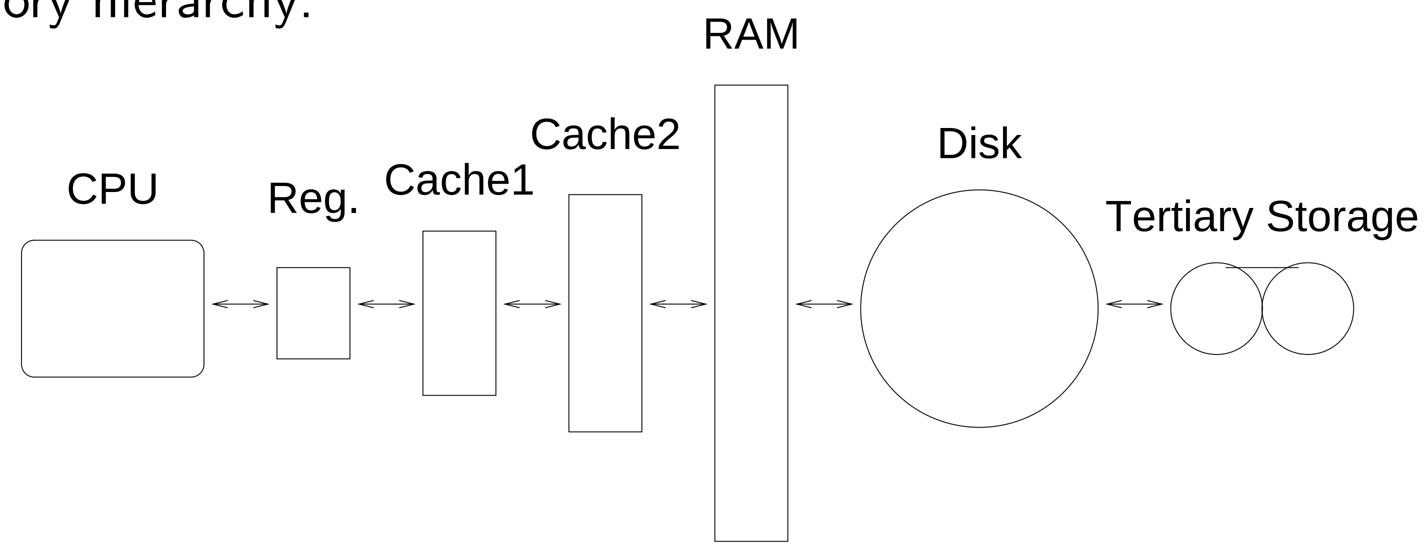
The standard model:



- ADD: 1 unit of time
- MULT: 1 unit of time
- BRANCH: 1 unit of time
- MEMACCESS: 1 unit of time

Reality

Memory hierarchy:



	Access time	*Volume*
Registers	1 cycle	1 Kb
Cache	5–10 cycles	1 Mb
RAM	50–100 cycles	1 Gb
Disk	30,000,000 cycles	250 Gb

CPU speed has improved faster than RAM access time and much faster than disk access time

Reality

Many real-life problems of **Terabyte** and even **Petabyte** size:

- weather
- geology/geograpy
- astrology
- financial
- WWW
- phone companies
- banks

I/O bottleneck

I/O is the bottleneck

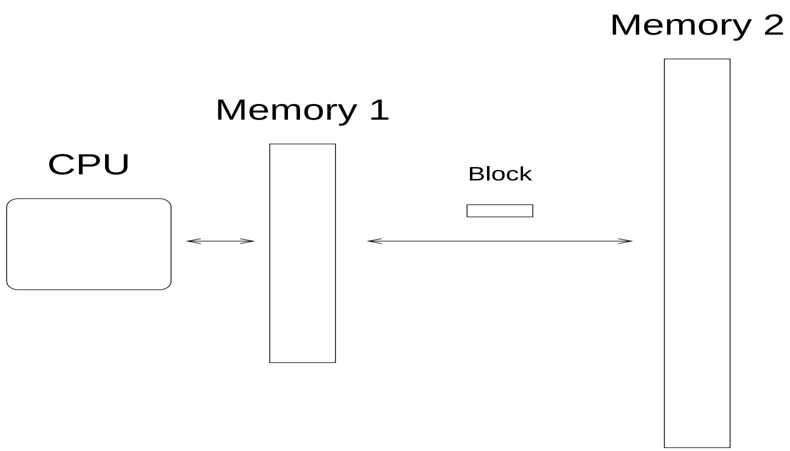
$\Downarrow$

I/O should be optimized (not instruction count)

We need new models for this.

Analysis of algorithms

New **I/O-model:**



Aggarwal, Vitter, 1988

Parameters:

N = no. of elements in problem.

M = no. of elements that fits in RAM.

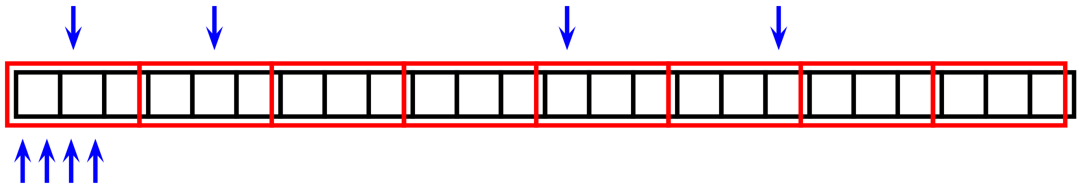
B = no. of elements in a block on disk.

Cost: Number of I/O's (block transfers) between Memory 1 and Memory 2.

Generic Example

Consider two $O(n)$ algorithms:

1. Memory accessed randomly $\Rightarrow$ page fault at each memory access.
2. Memory accessed sequentially $\Rightarrow$ page fault every B memory accesses.



$O(N)$ I/Os *vs.* $O(N/B)$ I/Os

Typically for disk: $B = 10^3 - 10^5$.

Note: 10^5 minutes $=$ 70 days, 10^5 days $=$ 274 years.

Specific Examples

Three $O(N \log N)$ CPU-time sorting algorithms:

	Worstcase	Inplace
QuickSort		$+$
MergeSort	$+$	
HeapSort	$+$	$+$

But:

QuickSort, MergeSort $\sim$ sequential access

HeapSort $\sim$ random access

So:

QuickSort:	$O(N\log_2(N/M)/B)$ I/Os
MergeSort:	$O(N\log_2(N/M)/B) I/Os$
HeapSort:	$O(N\log_2(N/M)) I/Os$

Course Contents

- The I/O model(s).
- Algorithms, data structures, and lower bounds for basic problems:
  - Permuting
  - Sorting
  - Searching
- I/O efficient algorithms and data structures for problems from
  - computational geometry,
  - strings,
  - graphs.

Along the way I: Generic principles for designing I/O-efficient algorithms.

Along the way II: Hands-on experience via projects.

Along the way III: Lots of beautiful algorithmic ideas.

Basic Results in the I/O-Model

Scanning: $\Theta(\frac{N}{B})$ I/Os

Sorting: $\Theta(\frac{N}{B} \log_{M/B}(\frac{N}{M}))$ I/Os

Permuting: $\Theta(\min\{N, \frac{N}{B} \log_{M/B}(\frac{N}{M})\})$ I/Os

Searching: $\Theta(\log_B(N))$ I/Os

Scanning, stacks, queues are I/O-efficient ($O(1/B)$ per operation) out of the box.

Most other algorithmic tasks need rethinking and new ideas.

Notable differences from standard internal model: linear time $= O(\frac{N}{B}) \neq O(N)$, sorting very close to linear time for normal parameters, sorting $=$ permuting for normal parameters, permuting $>$ linear time, sorting using search trees is far from optimal (search $>>$ sort/N).