Introduction to I/O Efficient Algorithms (External Memory Model)

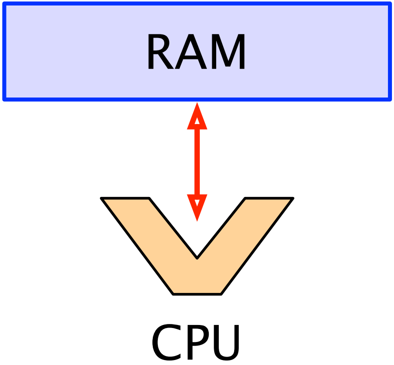
Jeff M. Phillips

August 29, 2011

Von Neumann Architecture

Model:

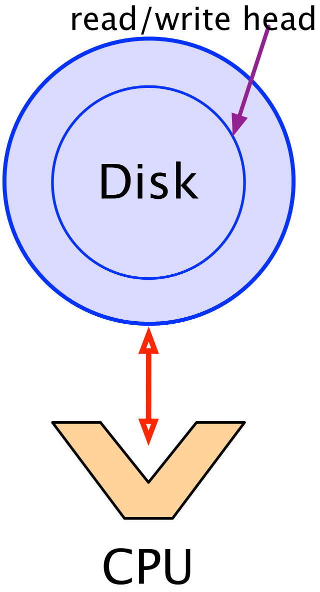
- CPU and Memory
- Read, Write, Operations $(+,\ -,\ *,\ \ldots)$ constant time
- polynomially equivalent to Turing Machine



Memory as Disk

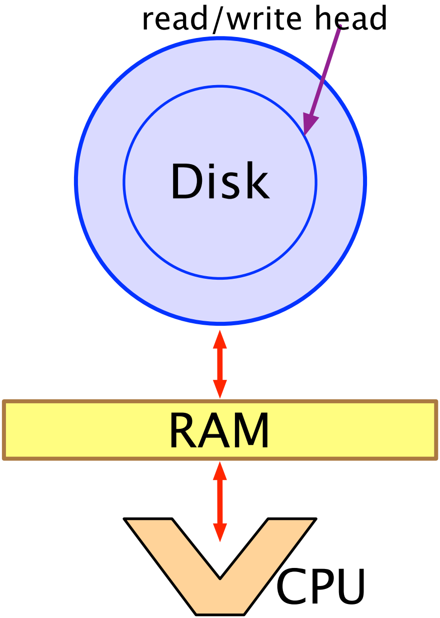
Reality:

- CPU and Memory
- CPU Operations ($+$, $-$, $*$, $\ldots$) constant time
- Read, Write not constant time (at least starting in 1980s).



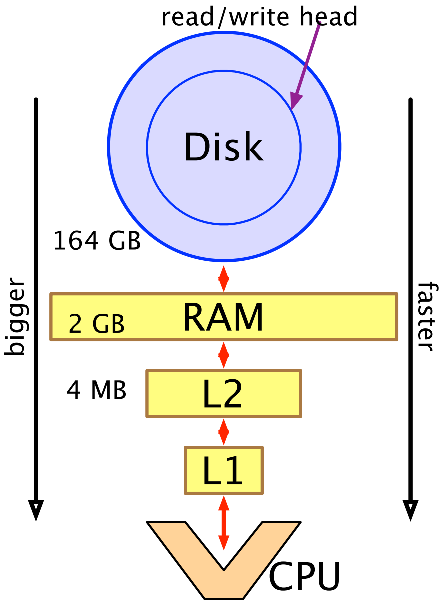
Cache

- through 1970s: cache access similar to memory access
- First commercially available 1982 (CP/M operating system)
- SmartDrive in Microsoft MS-DOS in 1988



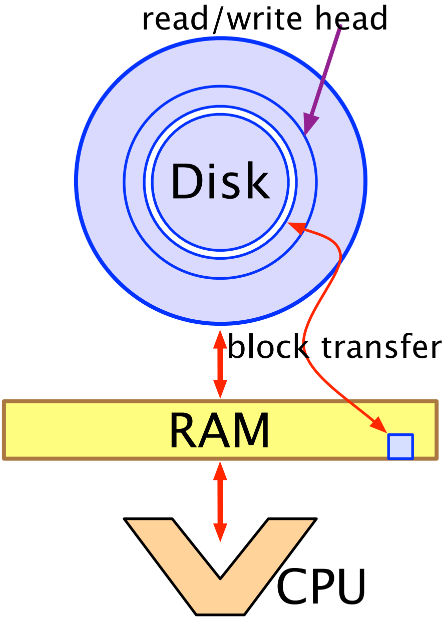
Memory Hierarchy

- 1980s $\dashrightarrow$ 1990s Hierarchy expanded
- 1989: 486 processor has L1 Cache in CPU had L2 off CPU on motherboard
- L2 popular as motherboard speed rose



Block Transfer

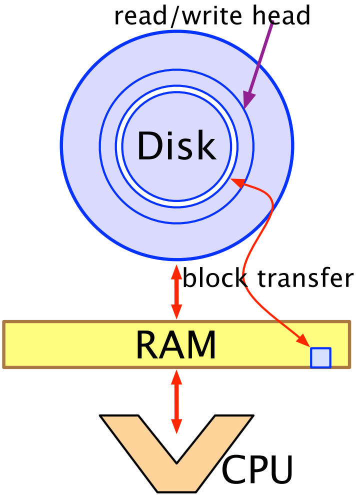
- Disk access is faster sequential: (B = 8-16KB)
- Sends whole block to RAM (size B).
- RAM has size $M > B^2$.
- Disk access is 10^6 more expensive than RAM access.
- Each block transfer is 1 I/O.
- Bound number of I/Os.



Block Transfer

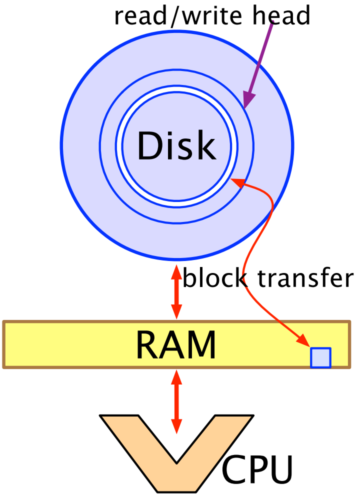
The difference in time between modern CPU and disk technologies is analogous to the difference in speed in sharpening a pencil using a sharpener on one's desk or by taking an airplane to the other side of the world and using a sharpener on someone else's desk.
- (Douglas Comer)

Scalability



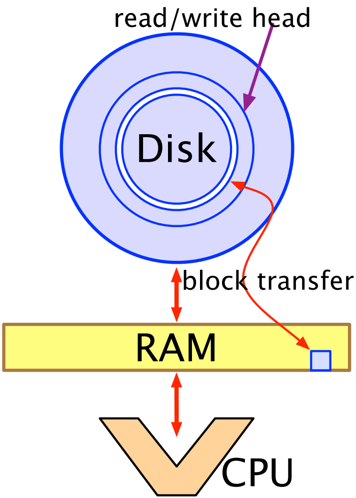
- Most programs developed in RAM model.
- Why don't they always thrash?

Scalability

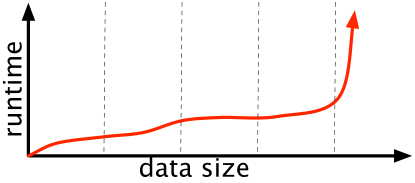


- Most programs developed in RAM model.
- Why don't they always thrash?
- Sophisticated OS shifts blocks under the hood (paging and prefetching).

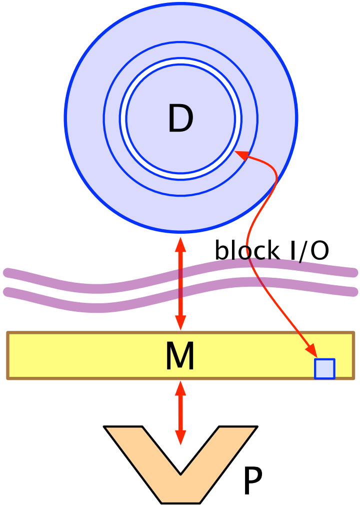
Scalability



- Most programs developed in RAM model.
- Why don't they always thrash?
- Sophisticated OS shifts blocks under the hood (paging and prefetching).
- Massive data and scattered access still spells doom.



External Memory Model



- N = size of problem instance
- B = size of disk block
- M = number of items that fits in Memory
- T = number of items in output
- I/O = block move between Memory and Disk

[Aggarwal and Vitter '88]
[Floyd '72]

Fundamental Bounds

	Internal	**External**
Scanning:	$O(N)$	

Fundamental Bounds

	Internal	**External**
Scanning:	$O(N)$	$O(N/B)$

Fundamental Bounds

	Internal	External
Scanning:	$O(N)$	$O(N/B)$
Sorting:	$O(N \log N)$	

Fundamental Bounds

	Internal	**External**
Scanning:	$O(N)$	$O(N/B)$
Sorting:	$O(N \log N)$	$O((N/B) \log_{M/B}(N/B))$

Fundamental Bounds

	Internal	External
Scanning:	$O(N)$	$O(N/B)$
Sorting:	$O(N \log N)$	$O((N/B) \log_{M/B}(N/B))$
Permuting:	$O(N)$	

Fundamental Bounds

	Internal	**External**
Scanning:	$O(N)$	$O(N/B)$
Sorting:	$O(N \log N)$	$O((N/B)\log_{M/B}(N/B))$
Permuting:	$O(N)$	$O(\min\{N, (N/B)\log_{M/B}(N/B)\})$

Fundamental Bounds

	Internal	External
Scanning:	$O(N)$	$O(N/B)$
Sorting:	$O(N \log N)$	$O((N/B) \log_{M/B}(N/B))$
Permuting:	$O(N)$	$O(\min\{N, (N/B) \log_{M/B}(N/B)\})$
Searching:	$O(\log_2 N)$	

Fundamental Bounds

	Internal	External
Scanning:	$O(N)$	$O(N/B)$
Sorting:	$O(N \log N)$	$O((N/B)\log_{M/B}(N/B))$
Permuting:	$O(N)$	$O(\min\{N, (N/B)\log_{M/B}(N/B)\})$
Searching:	$O(\log_2 N)$	$O(\log_B N)$

- Linear I/O: $O(N/B)$
- Permuting not linear
- Permuting and sorting equal (practically)
- B factor very important $\frac{N}{B} < \frac{N}{B}\log_{M/B}\frac{N}{B} \ll N$
- Cannot sorting optimally with search tree

Difference Between N and N/B

Consider traversing a linked list.

- Naive: $O(N)$ blocks, each hop to new block.
- Smart: $O(N/B)$ blocks, if sequential nodes in single block.

Difference Between N and N/B

Consider traversing a linked list.

- Naive: $O(N)$ blocks, each hop to new block.
- Smart: $O(N/B)$ blocks, if sequential nodes in single block.

Example: $N = 256 \times 10^6$, $B = 8000$, 1ms disk access time

- N I/Os takes 256×10^3 sec $= 4266$ min $= 71$ hours
- N/B I/Os takes $256/8$ sec $= 32$ sec

Attribution

These slides are heavily based on slides by Lars Arge
(a leading expert in the area of External Memory algorithms).
See: `http://www.daimi.au.dk/~large/ioS09/`