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

Jeff M. Phillips

August 30, 2013
Von Neumann Architecture

Model:
- CPU and Memory
- Read, Write, Operations (+, −, *, ...) constant time
- polynomially equivalent to Turing Machine
Memory as Disk

Reality:
- CPU and Memory
- CPU Operations (+, −, *, ...) 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 → 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: \( B = 8-16\text{KB} \)
- 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.
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?

Sophisticated OS shifts blocks under the hood (paging and prefetching).

Massive data and scattered access still spells doom.
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

<table>
<thead>
<tr>
<th></th>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scanning:</strong></td>
<td>$O(N)$</td>
<td></td>
</tr>
</tbody>
</table>

- Linear I/O: $O\left(\frac{N}{B}\right)$
- Permuting not linear
- Permuting and sorting equal (practically)
- $B$ factor very important
- $N_B < N_B \log M/B$

- $N_B \ll N_B$
Fundamental Bounds

<table>
<thead>
<tr>
<th>Scanning</th>
<th>Internal (O(N))</th>
<th>External (O(N/B))</th>
</tr>
</thead>
</table>

- **Linear I/O**: \(O(N/B)\)
- **Permuting not linear**
- **Permuting and sorting equal (practically)**
- **B factor very important**
- \(N/B < N/B \log M/B\)
- \(N/B \ll N/B\)
- **Cannot sort optimally with search tree**
Fundamental Bounds

<table>
<thead>
<tr>
<th>Activity</th>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanning</td>
<td>$O(N)$</td>
<td></td>
</tr>
<tr>
<td>Sorting</td>
<td>$O(N \log N)$</td>
<td>$O(N/B)$</td>
</tr>
<tr>
<td></td>
<td>Internal</td>
<td>External</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td><strong>Scanning:</strong></td>
<td>$O(N)$</td>
<td>$O(N/B)$</td>
</tr>
<tr>
<td><strong>Sorting:</strong></td>
<td>$O(N \log N)$</td>
<td>$O((N/B) \log_{M/B}(N/B))$</td>
</tr>
</tbody>
</table>
## Fundamental Bounds

<table>
<thead>
<tr>
<th></th>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanning</td>
<td>$O(N)$</td>
<td>$O(N/B)$</td>
</tr>
<tr>
<td>Sorting</td>
<td>$O(N \log N)$</td>
<td>$O((N/B) \log_{M/B}(N/B))$</td>
</tr>
<tr>
<td>Permuting</td>
<td>$O(N)$</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scanning:</strong></td>
<td>$O(N)$</td>
<td>$O(N/B)$</td>
</tr>
<tr>
<td><strong>Sorting:</strong></td>
<td>$O(N \log N)$</td>
<td>$O((N/B) \log_{M/B}(N/B))$</td>
</tr>
<tr>
<td><strong>Permuting:</strong></td>
<td>$O(N)$</td>
<td>$O(\min{N, (N/B) \log_{M/B}(N/B)})$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanning</td>
<td>$O(N)$</td>
<td>$O(N/B)$</td>
</tr>
<tr>
<td>Sorting</td>
<td>$O(N \log N)$</td>
<td>$O((N/B) \log_{M/B}(N/B))$</td>
</tr>
<tr>
<td>Permuting</td>
<td>$O(N)$</td>
<td>$O(\min{N, (N/B) \log_{M/B}(N/B)})$</td>
</tr>
<tr>
<td>Searching</td>
<td>$O(\log_2 N)$</td>
<td></td>
</tr>
</tbody>
</table>
Fundamental Bounds

<table>
<thead>
<tr>
<th></th>
<th>Internal</th>
<th>External</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scanning:</td>
<td>$O(N)$</td>
<td>$O(N/B)$</td>
</tr>
<tr>
<td>Sorting:</td>
<td>$O(N \log N)$</td>
<td>$O((N/B) \log_{M/B}(N/B))$</td>
</tr>
<tr>
<td>Permuting:</td>
<td>$O(N)$</td>
<td>$O(\min{N, (N/B) \log_{M/B}(N/B)})$</td>
</tr>
<tr>
<td>Searching:</td>
<td>$O(\log_2 N)$</td>
<td>$O(\log_B N)$</td>
</tr>
</tbody>
</table>

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

Templated Portable I/O Environment
Open source library of I/O-Efficient data structures.

- External memory merge sort
- B-Tree
- Priority queue
- Simple buffered stacks and queues

http://www.madalgo.au.dk/tpie/
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/