Massive Data Algorithmics

Gerth Stølting Brodal

University of Aarhus
Department of Computer Science

Danske Bank
Faglig Dag, January 17, 2008
The core problem...

![Graph showing running time vs. data size]

- Normal algorithm
- I/O-efficient algorithm

Main memory size
Outline of Talk

- Examples of massive data
- Hierarchical memory
- Basic I/O efficient techniques
- MADALGO center presentation
- A MADALGO project
Massive Data Examples

- Massive data being acquired/used everywhere
- Storage management software is billion-$ industry

More New Information Over Next 2 Years Than in All Previous History

- **Phone**: AT&T 20TB phone call database, wireless tracking
- **Consumer**: WalMart 70TB database, buying patterns
- **WEB**: Google index 8 billion web pages
- **Bank**: Danske Bank 250TB DB2
- **Geography**: NASA satellites generate Terrabytes each day
Massive Data Examples

- Society will become increasingly “data driven”
  - Sensors in building, cars, phones, goods, humans
  - More networked devices that both acquire and process data
    → Access/process data anywhere any time

- Nature 2/06 issue highlight trends in sciences:
  “2020 – Future of computing”
  - Exponential growth of scientific data
  - Due to e.g. large experiments, sensor networks, etc
  - Paradigm shift: *Science will be about mining data*
    → Computer science paramount in all sciences

- Increased data availability: “nano-technology-like” opportunity
Where does the slowdown come from?
Hierarchical Memory Basics

CPU

L1

L2

L3

RAM

Disk

Bottleneck

Increasing access time and space
Memory Hierarchy vs Running Time

running time

data size

L1  L2  L3  RAM
# Memory Access Times

<table>
<thead>
<tr>
<th>Memory Location</th>
<th>Latency</th>
<th>Relative to CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register</td>
<td>0.5 ns</td>
<td>1</td>
</tr>
<tr>
<td>L1 cache</td>
<td>0.5 ns</td>
<td>1-2</td>
</tr>
<tr>
<td>L2 cache</td>
<td>3 ns</td>
<td>2-7</td>
</tr>
<tr>
<td>DRAM</td>
<td>150 ns</td>
<td>80-200</td>
</tr>
<tr>
<td>TLB</td>
<td>500+ ns</td>
<td>200-2000</td>
</tr>
<tr>
<td>Disk</td>
<td>10 ms</td>
<td>$10^7$</td>
</tr>
</tbody>
</table>

Increasing
“The difference in speed 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.” (D. Comer)
Disk Mechanics

- I/O is often bottleneck when handling massive datasets
- Disk access is $10^7$ times slower than main memory access!
- Disk systems try to amortize large access time transferring large contiguous blocks of data
- Need to store and access data to take advantage of blocks!
The Algorithmic Challenge

- Modern hardware is not uniform — many different parameters
  - Number of memory levels
  - Cache sizes
  - Cache line/disk block sizes
  - Cache associativity
  - Cache replacement strategy
  - CPU/BUS/memory speed...

- Programs should ideally run for many different parameters
  - by knowing many of the parameters at runtime, or
  - by knowing few essential parameters, or
  - ignoring the memory hierarchies

- Programs are executed on unpredictable configurations
  - Generic portable and scalable software libraries
  - Code downloaded from the Internet, e.g. Java applets
  - Dynamic environments, e.g. multiple processes
Basic Algorithmic
I/O Efficient Techniques

- Scanning
- Sorting
- Recursion
- B-trees
I/O Efficient Scanning

sum = 0
for i = 1 to N do sum = sum + A[i]

\(O(N/B)\) I/Os
External-Memory Merging

Merging $k$ sequences with $N$ elements requires $O(N/B)$ I/Os (provided $k \leq M/B - 1$)
External-Memory Sorting

- MergeSort uses $O(N/B \cdot \log_{M/B}(N/B))$ I/Os
- Practice number I/Os: 4-6 x scanning input
B-trees - The Basic Searching Structure

- Searches
  Practice: 4-5 I/Os

- Repeated searching
  Practice: 1-2 I/Os

!!! Bottleneck !!!
Use sorting instead of B-tree (if possible)
About MADALGO (AU)

- Center of
- Lars Arge, Professor
- Gerth S. Brodal, Assoc. Prof.
- 3 PostDocs, 9 PhD students, 5 MSc students
- Total 5 year budget ~60 million kr (8M Euro)

- High level objectives
  - Advance algorithmic knowledge in massive data processing area
  - Train researchers in world-leading international environment
  - Be catalyst for multidisciplinary collaboration
Center Team

- International core team of algorithms researchers
- Including top ranked US and European groups
Center Collaboration

- COWI, DHI, DJF, DMU, Duke, NSCU
- Support from Danish Strategic Research Council and US Army Research Office
- Software platform for Galileo GPS
  - Various Danish academic/industry partners
  - Support from Danish High-Tech Foundation
- European massive data algorithmics network
  - 8 main European groups in area
MADALGO Focus Areas

- I/O Efficient Algorithms
- Streaming Algorithms
- Cache Oblivious Algorithms
- Algorithm Engineering
A MADALGO Project
Massive Terrain Data
Terrain Data

- New technologies:
  Much easier/cheaper to collect detailed data
- Previous ‘manual’ or radar based methods
  - Often 30 meter between data points
  - Sometimes 10 meter data available
- New laser scanning methods (LIDAR)
  - Less than 1 meter between data points
  - Centimeter accuracy (previous meter)

Denmark

- ~2 million points at 30 meter (<<1GB)
- ~18 billion points at 1 meter (>>1TB)
- COWI (and other) now scanning DK
- NC scanned after Hurricane Floyd in 1999
Hurricane Floyd

Sep. 15, 1999

7 am

3 pm
Denmark Flooding

+1 meter
+2 meter
Example: Terrain Flow

Conceptually flow is modeled using two basic attributes:

- **Flow direction**: The direction water flows at a point
- **Flow accumulation**: Amount of water flowing through a point

Flow accumulation used to compute other hydrological attributes: drainage network, topographic convergence index…
Example: Flow on Terrains

- Modeling of water flow on terrains has many important applications
  - Predict location of streams
  - Predict areas susceptible to floods
  - Compute watersheds
  - Predict erosion
  - Predict vegetation distribution
  - ……
Terrain Flow Accumulation

- Collaboration with environmental researchers at Duke University
  - Appalachian mountains dataset:
    - 800x800km at 100m resolution ⇒ a few Gigabytes
    - On ½GB machine: 14 days!!

- ArcGIS:
  - Performance somewhat unpredictable
  - Days on few gigabytes of data
  - Many gigabytes of data…..

- Appalachian dataset would be Terabytes sized at 1m resolution
Terrain Flow Accumulation: TerraFlow

- We developed theoretically I/O-optimal algorithms
- TPIE implementation was very efficient
  - Appalachian Mountains flow accumulation in 3 hours!

- Developed into comprehensive software package for flow computation on massive terrains: TerraFlow
  - **Efficient**: 2-1000 times faster than existing software
  - **Scalable**: >1 billion elements!
  - **Flexible**: Flexible flow modeling (direction) methods

- Extension to ArcGIS
Examples of Ongoing "Terrain Work"

- **Terrain modeling**, e.g.
  - “Raw” LIDAR to point conversion (LIDAR point classification) (incl feature, e.g. bridge, detection/removal)
  - Further improved flow and erosion modeling (e.g. carving)
  - Contour line extraction (incl. smoothing and simplification)
  - Terrain (and other) data fusion (incl format conversion)

- **Terrain analysis**, e.g.
  - Choke point, navigation, visibility, change detection, …

- **Major grand goal:**
  - Construction of hierarchical (simplified) DEM where derived features (water flow, drainage, choke points) are preserved/consistent
Summary

- Massive datasets appear everywhere
- Leads to scalability problems
  - Due to hierarchical memory and slow I/O
- I/O-efficient algorithms greatly improves scalability