Algorithms and Data Structures for Hierarchical Memory

Gerth Stølting Brodal
University of Aarhus
BRICS people

- Lars Arge, Professor  
  Joining September 1, 2004, SNF Rømer
- Gerth Stølting Brodal, Lektor  
  Carlsberg
- Rolf Fagerberg, Lektor  
  Since March 1, 2004, Odense
- Herman J. Haverkort, Post. Doc.  
  Joining October 1, 2004, SNF
- Gabriel Moruz, Ph.D. student

Former BRICS students with hierarchical memory research

- Lars Arge  
  Ph.D. 1996, Duke University
- Gerth Stølting Brodal  
  Ph.D. 1997, University of Aarhus
- Jakob Pagter  
  Ph.D. 2001, University of Aarhus
- Riko Jacob  
  Ph.D. 2002, ETH Zürich
Typical workstations...
Typical workstations...

families of
# Customizing a Dell 650

**May 26, 2004**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor speed</td>
<td>2.4 – 3.2 GHz</td>
</tr>
<tr>
<td>L3 cache size</td>
<td>0.5 – 2 MB</td>
</tr>
<tr>
<td>Memory</td>
<td>1/4 – 4 GB</td>
</tr>
<tr>
<td>Hard Disk</td>
<td>36 GB – 146 GB</td>
</tr>
<tr>
<td>CD/DVD</td>
<td>8 – 48x</td>
</tr>
<tr>
<td>L2 cache size</td>
<td>256 – 512 KB</td>
</tr>
<tr>
<td>L2 cache line size</td>
<td>128 Bytes</td>
</tr>
<tr>
<td>L1 cache line size</td>
<td>64 Bytes</td>
</tr>
<tr>
<td>L1 cache size</td>
<td>16 KB</td>
</tr>
</tbody>
</table>

**Hierarchical Memory**

[www.dell.dk](http://www.dell.dk)

[www.intel.com](http://www.intel.com)
Hierarchical Memory Basics

- Data moved between adjacent memory levels in blocks

Increasing access time and space
# More Hardware Specifications...

<table>
<thead>
<tr>
<th>Architecture type</th>
<th>Pentium 4</th>
<th>Pentium III</th>
<th>MIPS 10000</th>
<th>AMD Athlon</th>
<th>Itanium 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operation system</td>
<td>Modern CISC</td>
<td>Classic CISC</td>
<td>RISC</td>
<td>Modern CISC</td>
<td>EPIC</td>
</tr>
<tr>
<td>Clock rate</td>
<td>2400MHz</td>
<td>800MHz</td>
<td>175MHz</td>
<td>1333 MHz</td>
<td>1137 MHz</td>
</tr>
<tr>
<td>Address space</td>
<td>32 bit</td>
<td>32 bit</td>
<td>64 bit</td>
<td>32 bit</td>
<td>64 bit</td>
</tr>
<tr>
<td>Pipeline stages</td>
<td>20</td>
<td>12</td>
<td>6</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>L1 data cache size</td>
<td>8 KB</td>
<td>16 KB</td>
<td>32 KB</td>
<td>128 KB</td>
<td>32 KB</td>
</tr>
<tr>
<td>L1 line size</td>
<td>128 B</td>
<td>32 B</td>
<td>32 B</td>
<td>64 B</td>
<td>64 B</td>
</tr>
<tr>
<td>L1 associativity</td>
<td>4-way</td>
<td>4-way</td>
<td>2-way</td>
<td>2-way</td>
<td>4-way</td>
</tr>
<tr>
<td>L2 cache size</td>
<td>512 KB</td>
<td>256 KB</td>
<td>1024 KB</td>
<td>256 KB</td>
<td>256 KB</td>
</tr>
<tr>
<td>L2 line size</td>
<td>128 B</td>
<td>32 B</td>
<td>32 B</td>
<td>64 B</td>
<td>128 B</td>
</tr>
<tr>
<td>L2 associativity</td>
<td>8-way</td>
<td>4-way</td>
<td>2-way</td>
<td>8-way</td>
<td>8-way</td>
</tr>
<tr>
<td>TLB entries</td>
<td>128</td>
<td>64</td>
<td>64</td>
<td>40</td>
<td>128</td>
</tr>
<tr>
<td>TLB associativity</td>
<td>full</td>
<td>4-way</td>
<td>64-way</td>
<td>4-way</td>
<td>full</td>
</tr>
<tr>
<td>TLB miss handling</td>
<td>hardware</td>
<td>hardware</td>
<td>software</td>
<td>hardware</td>
<td>?</td>
</tr>
<tr>
<td>RAM size</td>
<td>512 MB</td>
<td>256 MB</td>
<td>128 MB</td>
<td>512 MB</td>
<td>3072 MB</td>
</tr>
</tbody>
</table>

**Hierarchical Memory**
Motivation

- Memory hierarchy has become a fact of life
- Accessing non-local storage may take a very long time
- Good locality is important to achieving high performance
- Handling massive data requires optimal memory usage

<table>
<thead>
<tr>
<th></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
Algorithmic Question

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

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

- Programs should ideally run for many different parameters
Algorithmic Question

- Modern hardware is not uniform — *many* different parameters
  - Number of caches
  - 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
  - by knowing few essential parameters
  - ignoring the memory hierarchies
Algorithmic Question

- Modern hardware is not uniform — many different parameters
  - Number of caches
  - 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
  - by knowing few essential parameters
  - ignoring the memory hierarchies

practice
Algorithmic Question

- Modern hardware is not uniform — many different parameters
  - Number of caches
  - 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
  - by knowing few essentiel parameters
  - 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
Hierarchical Memory Models

— many parameters

- Limited success since model too complicated

Increasing access time and space
**I/O Model** — two parameters

- Measure number of block transfers between two memory levels
- Bottleneck in many computations
- Very successful (+250 papers, many BRICS publications)
- Example: Sorting $N$ elements requires

\[
O \left( \frac{N}{B} \log_{M/B} \frac{N}{M} \right) \text{ I/Os}
\]

Aggarwal and Vitter 1988
I/O Model — two parameters

- Measure number of block transfers between two memory levels
- Bottleneck in many computations
- Very successful (+250 papers, many BRICS publications)
- Example: Sorting $N$ elements requires
  \[ O \left( \frac{N}{B} \log_{M/B} \frac{N}{M} \right) \] I/Os

Limitations

- Parameters $B$ and $M$ must be known
- Does not handle multiple memory levels
Cache Oblivious Model — no parameters!? 

Frigo, Leiserson, Prokop, Ramachandran 1999

- Program with only one memory
- Analyze in the I/O model for arbitrary $B$ and $M$
Cache Oblivious Model — no parameters!

Frigo, Leiserson, Prokop, Ramachandran 1999

- Program with only one memory
- Analyze in the I/O model for arbitrary $B$ and $M$

Advantages

- Optimal on arbitrary level $\Rightarrow$ optimal on all levels
- Portability
Engineering a Cache-Oblivious Sorting Algorithm, Brodal, Fagerberg, Vinther, 2004
Hierarchical Memory @ BRICS

- Ongoing research at BRICS since the start of BRICS
- Focus on foundational work for handling massive data sets
- Major research focus since 1998 (Brodal, Fagerberg)
- From September 2004 increased focus when Lars Arge (SNF Rømer) is joining BRICS
- BRICS publications in leading theoretical computer science conference proceedings
- Several surveys and book chapters on algorithms for massive data sets / external memory algorithms / cache-oblivious algorithms by BRICS authors
- EEF summer school on Massive data sets (2002)