Algorithms and Computation in Signal Processing

special topic course 18-799B
spring 2005
18th Lecture Mar. 17, 2005

Instructor: Markus Pueschel
TA: Srinivas Chellappa
Summary of Insights so Far
## Comparison Atlas, Sparsity, FFTW

<table>
<thead>
<tr>
<th></th>
<th>Atlas (MMM)</th>
<th>Sparsity/Bebop (sparse MVM)</th>
<th>FFTW</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cache optimization</strong>&lt;br&gt;(locality)</td>
<td>Blocking</td>
<td>Blocking (rarely useful)</td>
<td>recursive FFT, fusion of steps</td>
</tr>
<tr>
<td><strong>Register optimization</strong>&lt;br&gt;(locality)</td>
<td>Blocking</td>
<td>Blocking (determines sparse format)</td>
<td>Scheduling of small FFTs</td>
</tr>
<tr>
<td><strong>Optimized basic blocks</strong></td>
<td>Unrolling, instruction ordering, scalar replacement, simplifications (for FFT)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Other optimizations</strong></td>
<td></td>
<td></td>
<td>Precomputation of constants</td>
</tr>
<tr>
<td><strong>Adaptivity</strong></td>
<td>Search over blocking parameters</td>
<td>Search over register blocking size</td>
<td>Search over recursion strategy</td>
</tr>
</tbody>
</table>
Guideline for writing fast Code (I)

- **Avoid obvious mistakes**
  - Know the available algorithms: use good algorithms
  - Precompute once were possible (e.g., constants)
  - Give the compiler a chance: write simple code, avoid complicated data structures
  - Understand were the runtime is wasted: code profiling
  - Use good compiler flags, try alternatives

- **Optimization for caches**
  - Recursive is better then iterative
  - Understand your code in terms of cache behavior and try to improve
  - Know your cache size and maybe other parameter
Guideline for writing fast Code (II)

- **Basic block optimization**
  - For the innermost kernels use unrolled code: no loops, recursive calls or other control structures
  - Order instructions for register locality and/or instruction parallelism; scalar replacement for variables being reused; other optimizations
  - Maybe: check assembly code

- **Adaptivity through search over alternatives**
  - Accept that you can’t know the right answers for all choices
  - Search over a relevant subset of possible algorithms and/or implementation options

- **After optimization check whether you still use the right algorithm**
Convolution/Filtering

- Circular convolution
- Linear convolution
- Correlation


- Blackboard