Java Signal Processing: FFT’s with Bytecodes

John Glossner, Jesse Thilo and Stamatis Vassiliadis

Delft University of Technology
CARDIT - Computer Architecture and Digital Technique
Delft, The Netherlands
{glossner,jthilo,stamatis}@cardit.et.tudelft.nl
Overview

- Java Properties
- Tensor Algebra
- FFT
- Experiments
- Results
- Conclusions
Java Properties

- Object-Oriented Programming Language
- Programmer Supplied Parallelism (Threads)
- Dynamically Linked
- Garbage Collected
- Strongly Typed
  - Statically determinable type state enables simple on-the-fly translation of bytecodes into efficient machine code [Gos95]
- Compiled to Platform Independent Virtual Machine
Tensor Product

Tensor Product Definition

\[ A_{t_1, e_1} \otimes B_{t_2, e_2} = \begin{bmatrix}
    a_{00}B & \ldots & \ldots & a_{0(c_1-1)}B \\
    a_{10}B & \ddots & \ddots & \vdots \\
    \vdots & \ddots & \ddots & \vdots \\
    a_{(t_1-1)0}B & \ldots & \ldots & a_{(t_1-1)(c_1-1)}B
\end{bmatrix} \]

Tensor Permutation Matrix \( P_{t}^{tc} \) is generated by:

\( (P_{t}^{tc})_{i,j} = 1 \) for \( i = j = (tc - 1) \) and \( j = it \mod (tc - 1) \) for \( 0 \leq i < (tc - 1) \)

\( = 0 \) otherwise

Tensor Commutation

\[ A_{t} \otimes B_{c} = P_{t}^{tc} (B_{c} \otimes A_{t}) P_{c}^{tc} \]
Tensor FFT

\[ FFT(x) = y_k = \sum_{j=0}^{N-1} \omega^{jk} x_j \quad \text{for} \ 0 \leq k < N \quad \text{where} \ \omega = e^{\frac{-2\pi i}{N}}, \ i = \sqrt{-1} \]

\[ F_n = F_{tc} = P_t^n (I_c \otimes F_t) P_c^n T_c^n (I_t \otimes F_c) P_t^n \quad \text{where} \quad F_n = \omega^{jk} \]

\[ T_n = \text{diag} (I_c, D_c(n),...,D_c(n)^{t-1}) \]

\[ D_c(n) = \text{diag} (1, \omega,...,\omega^{c-1}) \]
Experiments

- **Traditional FFT**
  - C algorithm from [Press92]
  - Multiple Sizes

- **Hand Translated to Java**
  - Does not use Java specific capabilities

- **Comparison**
  - gcc 2.7.2
  - Matlab built-in
  - Java Interpreted
  - Java JIT
  - Toba off-line compiler

- **Tensor FFT**
  - Parallel Pease Implementation
  - Optimized for MT/MP Applications

- **Coded In Matlab and Java**
  - Uses Language Specific Constructs
  - Matlab: Vectorized, built-in Complex
  - Java: Objects, Complex Class
Traditional FFT Results

<table>
<thead>
<tr>
<th></th>
<th>F4</th>
<th>F16</th>
<th>F64</th>
<th>F256</th>
<th>F1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc -O3</td>
<td>4.8</td>
<td>12</td>
<td>37</td>
<td>149</td>
<td>642</td>
</tr>
<tr>
<td>Solaris 2.6 w/JIT</td>
<td>5.8</td>
<td>20</td>
<td>81</td>
<td>384</td>
<td>1780</td>
</tr>
<tr>
<td>Toba 1.0b6 -O3</td>
<td>6.3</td>
<td>22</td>
<td>93</td>
<td>591</td>
<td>2070</td>
</tr>
<tr>
<td>gcc unopt</td>
<td>8.0</td>
<td>30</td>
<td>132</td>
<td>630</td>
<td>2890</td>
</tr>
<tr>
<td>Kaffe 0.9.2</td>
<td>10.2</td>
<td>41</td>
<td>185</td>
<td>867</td>
<td>4070</td>
</tr>
<tr>
<td>Matlab built-in</td>
<td>32.0</td>
<td>60</td>
<td>180</td>
<td>300</td>
<td>1300</td>
</tr>
<tr>
<td>Java 1.1.4 Interp</td>
<td>34.7</td>
<td>181</td>
<td>918</td>
<td>4520</td>
<td>22000</td>
</tr>
</tbody>
</table>
Traditional FFT Rel. Results

Relative Time

<table>
<thead>
<tr>
<th></th>
<th>F4</th>
<th>F16</th>
<th>F64</th>
<th>F256</th>
<th>F1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcc -O3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Solaris 2.6 w/JIT</td>
<td>1.2</td>
<td>1.6</td>
<td>2.2</td>
<td>2.6</td>
<td>2.8</td>
</tr>
<tr>
<td>Toba 1.0b6 -O3</td>
<td>1.3</td>
<td>1.8</td>
<td>2.5</td>
<td>4.0</td>
<td>3.2</td>
</tr>
<tr>
<td>gcc unopt</td>
<td>1.7</td>
<td>2.5</td>
<td>3.6</td>
<td>4.2</td>
<td>2.5</td>
</tr>
<tr>
<td>Kaffe 0.9.2</td>
<td>2.1</td>
<td>3.3</td>
<td>5.0</td>
<td>5.8</td>
<td>6.3</td>
</tr>
<tr>
<td>Matlab built-in</td>
<td>6.7</td>
<td>4.9</td>
<td>4.9</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Java 1.1.4 Interp</td>
<td>7.2</td>
<td>15.0</td>
<td>25.0</td>
<td>30.0</td>
<td>34.0</td>
</tr>
</tbody>
</table>
## Tensor FFT Time

<table>
<thead>
<tr>
<th></th>
<th>F4</th>
<th>F16</th>
<th>F64</th>
<th>F256</th>
<th>F1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solaris 2.6 w/JIT</td>
<td>44</td>
<td>1000</td>
<td>7200</td>
<td>41000</td>
<td>221000</td>
</tr>
<tr>
<td>Toba 1.0b6 -O3</td>
<td>28</td>
<td>730</td>
<td>5100</td>
<td>29000</td>
<td>166000</td>
</tr>
<tr>
<td>Java 1.1.4 Interp</td>
<td>46</td>
<td>1200</td>
<td>8400</td>
<td>47000</td>
<td>254000</td>
</tr>
<tr>
<td>Kaffe 0.9.2</td>
<td>81</td>
<td>1900</td>
<td>15000</td>
<td>196000</td>
<td>629000</td>
</tr>
<tr>
<td>Matlab Interp</td>
<td>250</td>
<td>7100</td>
<td>50000</td>
<td>318000</td>
<td>3470000</td>
</tr>
</tbody>
</table>
# Tensor Relative Time

<table>
<thead>
<tr>
<th></th>
<th>F4</th>
<th>F16</th>
<th>F64</th>
<th>F256</th>
<th>F1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solaris 2.6 w/JIT</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Toba 1.0b6 -O3</td>
<td>0.6</td>
<td>0.7</td>
<td>0.7</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>Java 1.1.4 Interp</td>
<td>1.1</td>
<td>1.2</td>
<td>1.2</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>Kaffe 0.9.2</td>
<td>1.8</td>
<td>1.9</td>
<td>2.1</td>
<td>4.8</td>
<td>2.8</td>
</tr>
<tr>
<td>Matlab built-in</td>
<td>5.7</td>
<td>7.1</td>
<td>6.9</td>
<td>7.8</td>
<td>15.7</td>
</tr>
</tbody>
</table>
Conclusions

- **Java May Offer Sufficient FFT Performance**
  - Small FFT’s are within 20% to 60% of C performance
  - Larger FFT’s are 2-3x less efficient
  - Object Oriented FFT’s are much less efficient (>>30x)
  - Better Java Compiler Technology may decrease this gap
  - Direct Execution on Java Processors may provide even better performance

- **Matlab Easier To Program But Java Executes 6-8x Faster**
  - Built-in Complex Type Eases Matlab Programming
  - Easy Integration With Other Built-in Matlab routines
  - Matlab native C compiler may decrease this gap

- **JIT Compilers Offer Significant Performance Benefits On C-like Java**
  - Traditional FFT speedup of 6-12x over Interpreted Java
  - Object Creation (e.g. garbage collection) obviates improvements
    - The Toba off-line compiler provides the best performance when many objects are created
  - JIT technology is still in infancy