Outline

• Introduction to Jacket for MATLAB®
• GFOR
• Comparison with GPU-PCT alternative
• Case Studies: Genomics Examples
Matrix Types

gdouble  
  double precision

gsingle  
  single precision

glogical  
  boolean

guint#  
  unsigned integers

gint#  
  integers
Matrix Types: ND Support

- Vectors
- Matrices
- Volumes
- ... ND
Matrix Types: Easy Manipulation

A(1,1)  A(1,:)  A(:,1)  A(:,:,2)
n = 20e6;  % 20 million random samples
X = grand(1,n,'gdouble');
Y = grand(1,n,'gdouble');
distance_to_origin = sqrt( X.*X + Y.*Y );
is_inside = (distance_to_origin <= 1);
pi = 4 * sum(is_inside) / n;
n = 20e6;  % 20 million random samples
X = grand(1,n,'gdouble');
Y = grand(1,n,'gdouble');

distance_to_origin = sqrt( X.*X + Y.*Y );
is_inside = (distance_to_origin <= 1);
pi = 4 * sum(is_inside) / n;
Easy GPU Acceleration of M code

No GPU-specific stuff involved (no kernels, no threads, no blocks, just regular M code)

“Very little recoding was needed to promote our Lattice Boltzmann Model code to run on the GPU.” –Dr. Kevin Tubbs, HPTi
Easy Multi GPU Scaling

```matlab
y = gzeros( 5, 5, n );
for i = 1:n,
    gselect(i);
    x = grand(5,5);
    y(:,:,i) = fft(x); % more work in queue
end

% all GPUs are now computing simultaneously, until done
```

% choose GPU for this iteration
% add work to GPU’s queue
Technology Stack

• A full system making optimizations for you
• Including
  – “Core” runtime
  – “JIT” smart copy/exec
  – “Calls” functionality
Automated Optimizations

\[ A = \sin( x + y )^2 \]
Automated Optimizations

\[ A = \sin(x + y)^2 \]
GFOR

GPU FOR-loops
GFOR – Parallel FOR-loop for GPUs

• Like a normal FOR-loop, but faster

Regular FOR-loop (3 serial kernel launches)

```plaintext
for i = 1:3
    C(:, :, i) = A(:, :, i) * B;
```

Parallel GPU FOR-loop (only 1 kernel launch)

```plaintext
gfor i = 1:3
    C(:, :, i) = A(:, :, i) * B;
```
Example: Matrix Multiply

Regular FOR-loop (3 serial kernel launches)

```matlab
for i = 1:3
    C(:,:,i) = A(:,:,i) * B;
```

iteration $i = 1$

$C(:,:,i) = A(:,:,i) * B$
Example: Matrix Multiply

Regular FOR-loop (3 serial kernel launches)

\[ \text{for } i = 1:3 \]
\[ C(:,:,i) = A(:,:,i) \ast B; \]

iteration \( i = 1 \)

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\ast
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
= 
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

iteration \( i = 2 \)

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\ast
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
= 
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]
Example: Matrix Multiply

Regular FOR-loop (3 serial kernel launches)

```
for i = 1:3
    C(:,:,i) = A(:,:,i) * B;
```

Iteration i = 1

\[
\begin{bmatrix}
C(:,:,1) \\
A(:,:,1)
\end{bmatrix} = \begin{bmatrix}
\end{bmatrix} * \begin{bmatrix}
B
\end{bmatrix}
\]

Iteration i = 2

\[
\begin{bmatrix}
C(:,:,2) \\
A(:,:,2)
\end{bmatrix} = \begin{bmatrix}
\end{bmatrix} * \begin{bmatrix}
B
\end{bmatrix}
\]

Iteration i = 3

\[
\begin{bmatrix}
C(:,:,3) \\
A(:,:,3)
\end{bmatrix} = \begin{bmatrix}
\end{bmatrix} * \begin{bmatrix}
B
\end{bmatrix}
\]
Example: Matrix Multiply

Parallel GPU FOR-loop (only 1 kernel launch)

\[ \text{gfor} \ i = 1:3 \]
\[ C(:,:,1:3) = A(:,:,1:3) \times B; \]

simultaneous iterations \( i = 1:3 \)
Example: Matrix Multiply

Parallel GPU FOR-loop (only 1 kernel launch)

```matlab
for i = 1:3
    C(:,:,i) = A(:,:,i) * B;
end
```

Simultaneous iterations $i = 1:3$
Example: Summing over Columns

• Think of gfor as “syntactic sugar” to write vectorized code in an iterative style.

Three passes to sum all columns of B

for i = 1:3
    A(i) = sum(B(:,i));
end

One pass to sum all columns of B

gfor i = 1:3
    A(i) = sum(B(:,i));
end

Both equivalent to “\texttt{sum(B)}”, but latter is faster (more explicitly written)
Jacket versus PCT

parallel computing toolbox

MATLAB and PCT are products and trademarks of MathWorks.
Compare with R2012a PCT

• Jacket is faster
• Jacket does not use Java
• Jacket is very mature (~5 years)
• Jacket includes more functionality

MATLAB and PCT are products and trademarks of MathWorks.
# Speedups with Jacket

<table>
<thead>
<tr>
<th>Operation</th>
<th>Jacket speedup over R2012A PCT™</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic: $1 + 2\cdot x + 3\cdot x.\cdot^2 + 4\cdot x.\cdot^3 + 5\cdot x.\cdot^4$</td>
<td>109x</td>
</tr>
<tr>
<td>Trigonometric Functions: \texttt{cos}</td>
<td>76x</td>
</tr>
<tr>
<td>Sorting: \texttt{sort}</td>
<td>41x</td>
</tr>
<tr>
<td>Convolution 2D: \texttt{conv2}</td>
<td>29x</td>
</tr>
<tr>
<td>Elementwise Multiplication: \texttt{times}</td>
<td>19x</td>
</tr>
<tr>
<td>Data Manipulation: \texttt{reshape}</td>
<td>19x</td>
</tr>
<tr>
<td>Exponential Power: \texttt{power}</td>
<td>11x</td>
</tr>
<tr>
<td>Linear Algebra: \texttt{transpose}</td>
<td>5x</td>
</tr>
<tr>
<td>Convolution 1D: \texttt{conv}</td>
<td>3x</td>
</tr>
<tr>
<td>Search: \texttt{find}</td>
<td>3x</td>
</tr>
<tr>
<td>Reductions: \texttt{sum, min, max}</td>
<td>2x</td>
</tr>
<tr>
<td>Linear Algebra: matrix inverse, \texttt{lu, qr, mldivide}</td>
<td>1.4x</td>
</tr>
</tbody>
</table>
Jacket has 10X more functions...

<table>
<thead>
<tr>
<th>Category</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>gfor (loops)</td>
<td>reductions</td>
</tr>
<tr>
<td></td>
<td>sum, min, max, any, all, nnz, prod</td>
</tr>
<tr>
<td></td>
<td>vectors, columns, rows, etc</td>
</tr>
<tr>
<td>dense linear algebra</td>
<td>convolutions</td>
</tr>
<tr>
<td></td>
<td>2D, 3D, ND</td>
</tr>
<tr>
<td>FFTs</td>
<td>interp and rescale</td>
</tr>
<tr>
<td></td>
<td>vectors, matrices            rescaling</td>
</tr>
<tr>
<td>image processing</td>
<td>sorting</td>
</tr>
<tr>
<td></td>
<td>along any dimension rescaling</td>
</tr>
<tr>
<td></td>
<td>find</td>
</tr>
<tr>
<td>help</td>
<td>gprofview</td>
</tr>
</tbody>
</table>

and many more...
Easy To Maintain

• Write your code once and let Jacket carry you through the coming hardware evolution.
  – Each new Jacket release improves the speed of your code, without any code modification.
  – Each new Jacket release leverages latest GPU hardware (e.g. Fermi, Kepler), without any code modification.
Case Studies
17X Neuro-imaging Georgia Tech
20X Bio-Research CDC
20X Video Processing Google
45X Radar Imaging System Planning
12X Medical Devices Spencer Tech

5X Weather Modeling NCAR
35X Power Engineering IIT India
17X Track Bad Guys BAE Systems
70X Drug Delivery Georgia Tech
35X Bioinformatics Leibniz
Case Study: CDC Genomics

- Hepatitis C Virus (HCV)
- Goal to explore random genetic mutations
- 10,000 random alignments
  - simulating the distribution of correlation values under the null hypothesis that substitutions of amino acids at two sites are statistically independent (how aa’s mutate HCV)
Case Study: CDC Genomics

- 10,000 random alignments intractable on CPU
- Addition of GPUs brings ~18X speedup
Case Study: Leibniz Institute

- High Throughput Multi-Dimensional Scaling (HiT-MDS)
- High dimensional data reconstructed for visualization
- Goal to understand data
- Speedup: 35X
Case Study: Spencer Technologies

- Real-time signal processing
- 64 ultrasound sensors
- Precise brain blood flow
- Speedup: 12X
Discussion

Faster MATLAB® through GPU computing