Dancing Monkeys: Accelerated

GPU-Accelerated Beat Detection
for *Dancing Monkeys*

Philip Peng, Yanjie Feng
UPenn CIS 565 Spring 2012

Final Project - Midpoint Presentation

img src: http://www.dcrblogs.com/wp-content/uploads/2010/03/radioactive-dancing-monkeys-fastest-ani.gif

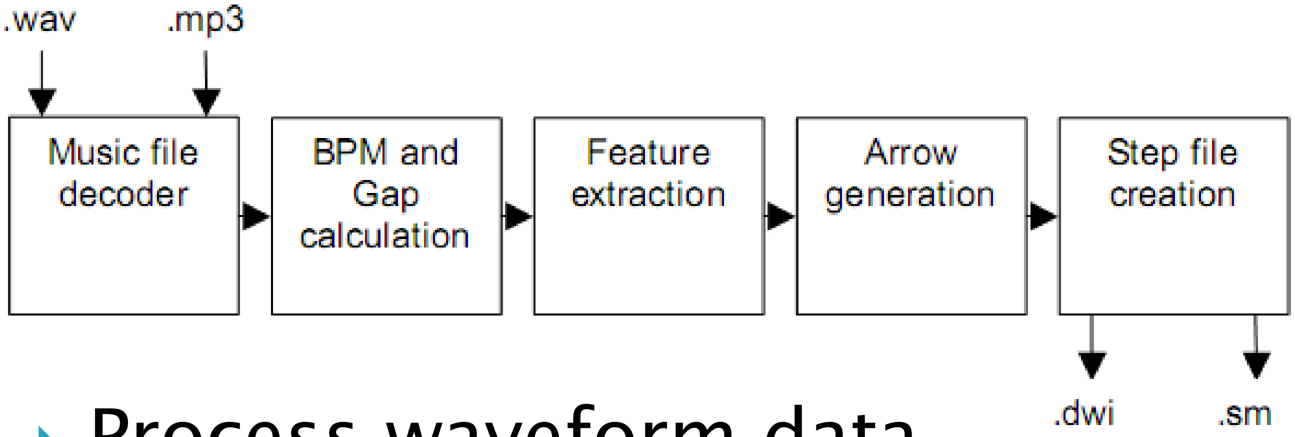
Project Description

- Dancing Monkeys
  - Create DDR step patterns from arbitrary songs
  - Highly precise beat detection algorithm (accurate within <0.0001 BPM)
  - Nov 1, 2003 by Karl O'Keeffe
  - MATLAB program, CC license
  - http://monket.net/dancing-monkeys-v2/

- GPU Acceleration
  - Algorithm used = brute force BPM comparisons
  - GPUs are good with parallel number crunching!

img src: http://monket.net/uploaded-v2/Feet.png

Dancing Monkeys Architecture



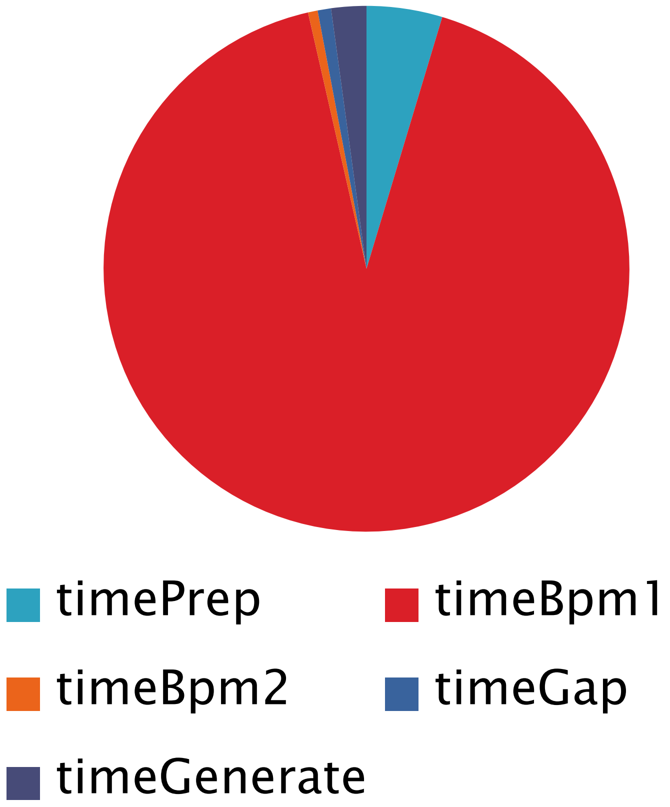
- Process waveform data
- Calculate BPM (first pass)
- Calculate BPM (second pass)
- Calculate gap time
- Generate arrow patterns from waveform data

Timing Breakdown

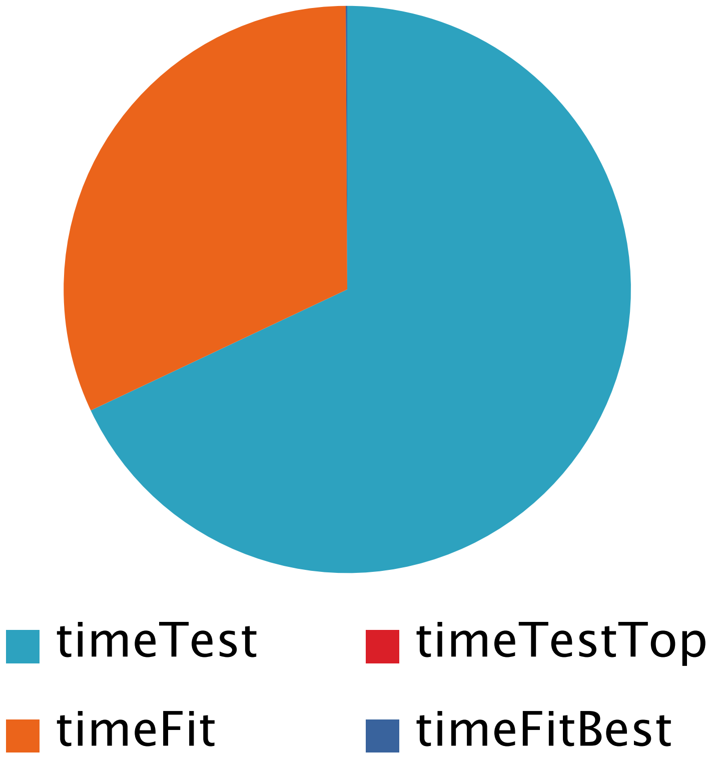
Process	Time (s)
timeProgram	**202.748426**
timeArgs	0.082273
timeSong	202.651163
timePrep	**9.432683**
timeInfo	7.371580
timeData	1.797109
timePeaks	0.253459
timeBpm1	**185.960192**
timeTest	126.409699
timeTestTop	0.002460
timeFit	59.377492
timeFitBest	0.154692
timeBpm2	**1.200393**
timeTest	1.184987
timeTestTop	0.000064
timeFit	0.000122
timeFitBest	0.006139
timeGap	**1.663195**
timeEnergy	0.040256
timeSimilar	1.617153
timeGenerate	**4.375886**
timeCliques	0.035418
timePause	0.431286
timeArrow	0.350256
timeOutput	3.546520

Timing Breakdown

timeSong Breakdown

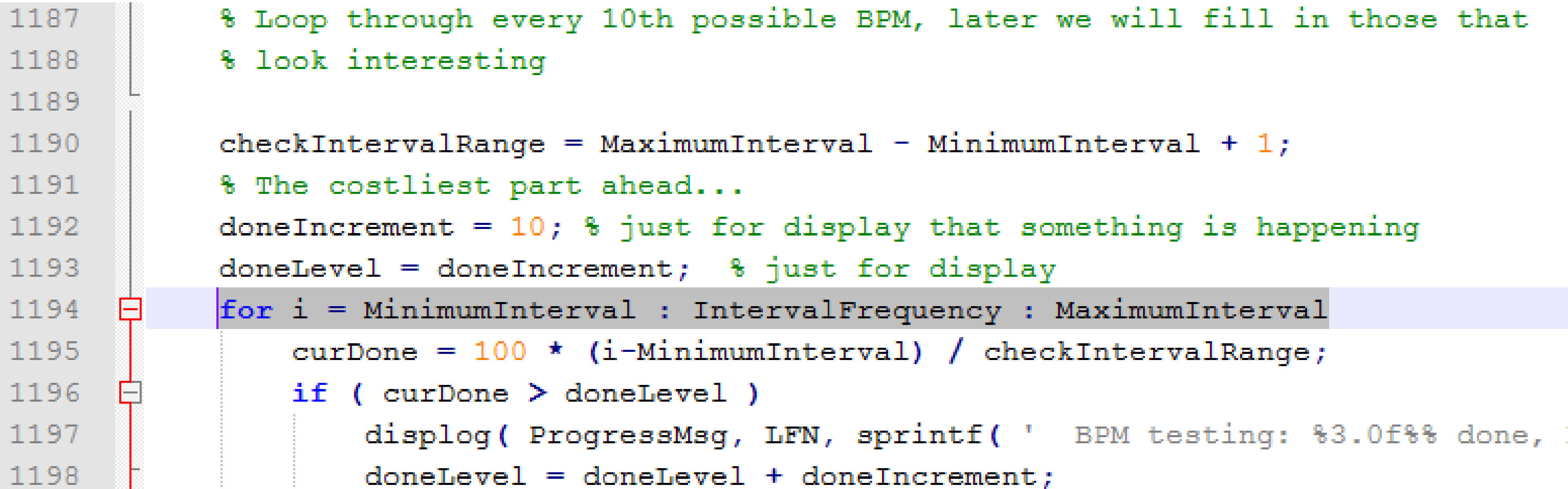


timeBpm Breakdown



Code Analysis

- timeBPM (first pass) longest: brute force BPM comparisons
  - BPM [89, 205], Frequency = 44100
  - Interval = round(Frequency / (BPM / 60));
  - Interval = [12907, 29730], IntervalFrequency = 10
  - Total of **1682 loops**



```
% Loop through every 10th possible BPM, later we will fill in those that
% look interesting

checkIntervalRange = MaximumInterval - MinimumInterval + 1;
% The costliest part ahead...
doneIncrement = 10; % just for display that something is happening
doneLevel = doneIncrement;  % just for display
for i = MinimumInterval : IntervalFrequency : MaximumInterval
    curDone = 100 * (i-MinimumInterval) / checkIntervalRange;
    if ( curDone > doneLevel )
        displog( ProgressMsg, LFN, sprintf( '  BPM testing: %3.0f%% done, 
        doneLevel = doneLevel + doneIncrement;
```

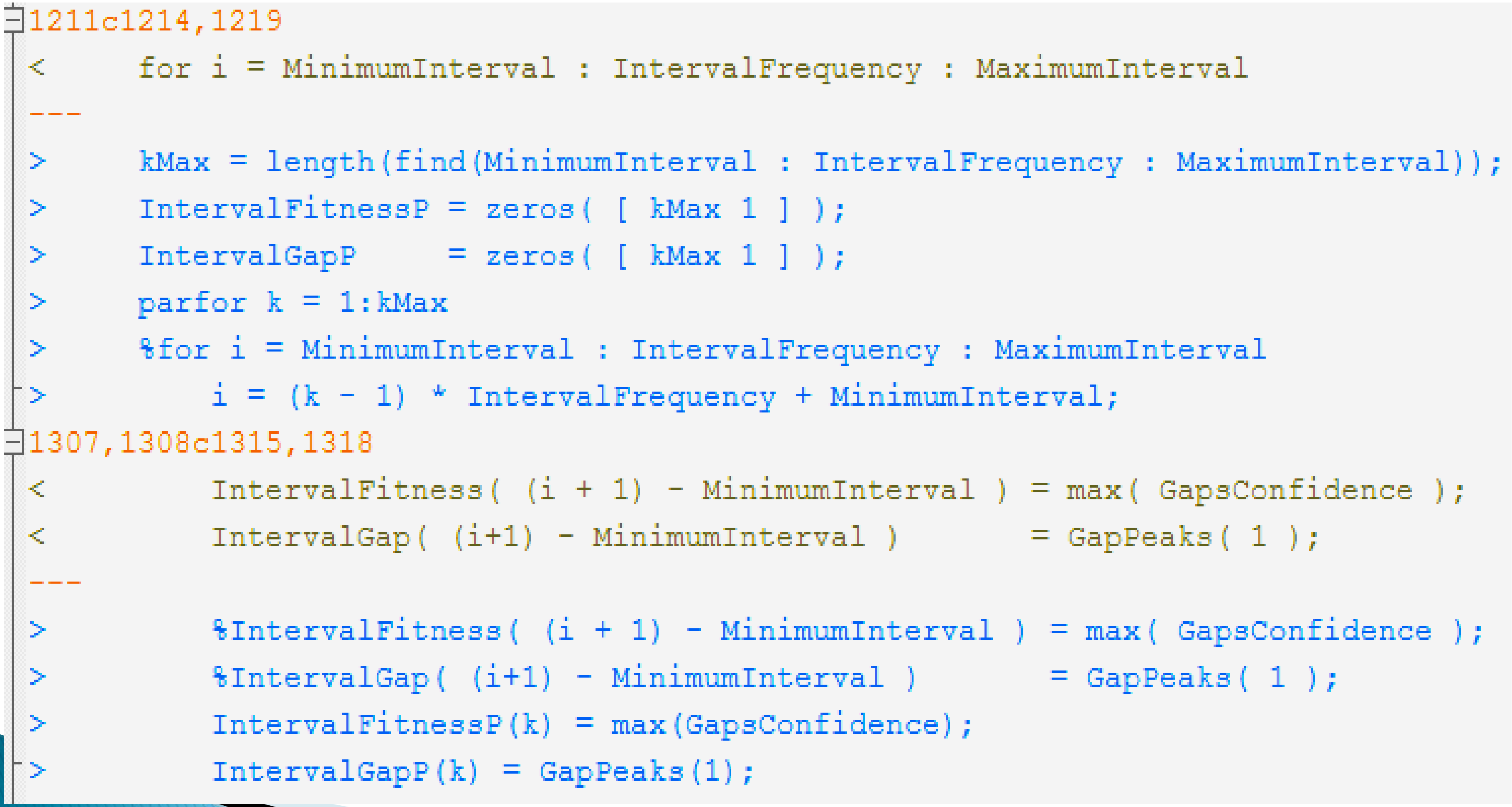
CPU Parallelization – Approach

- MATLAB’s Parallel Computing Toolbox
- Replace *for* loops with MATLAB’s *parfor*
  - Run loop in parallel, one per CPU core
  - http://www.mathworks.com/help/toolbox/distcomp/parfor.html
- Require code modification
  - matlabpool
  - Temporary arrays
  - Index recalculations

MathWorks

Parallel Computing Toolbox™

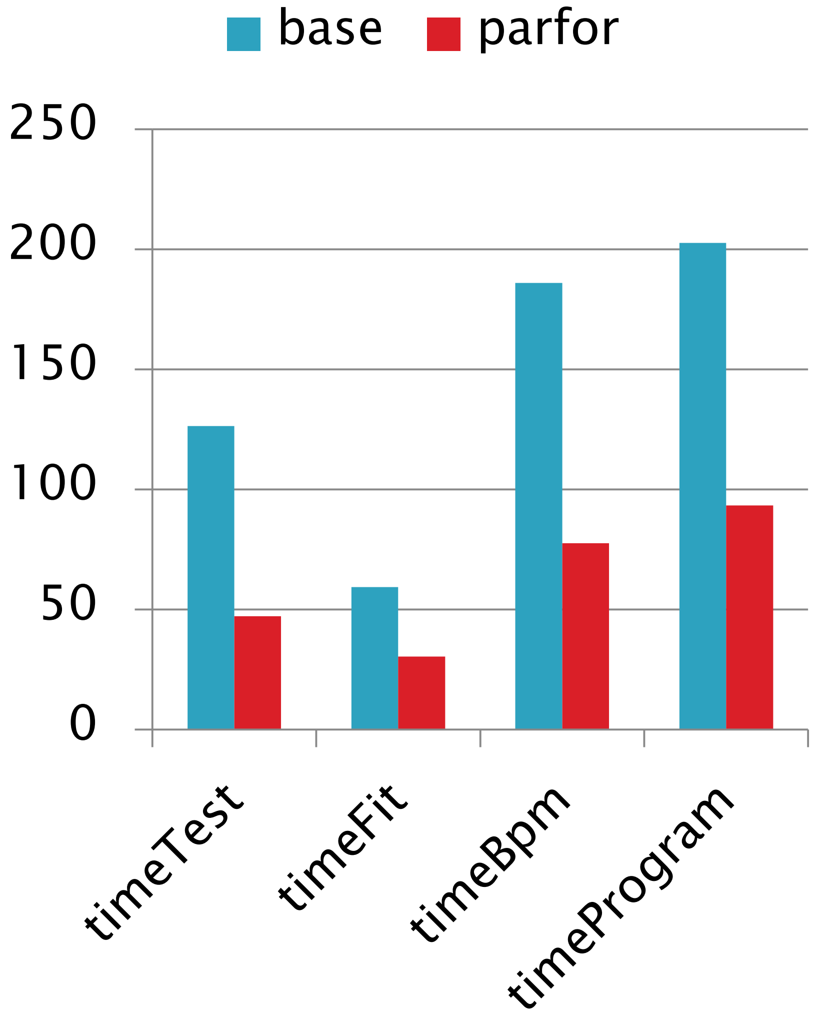
CPU Parallelization – Code



```
1211c1214,1219
<     for i = MinimumInterval : IntervalFrequency : MaximumInterval
---
>     kMax = length(find(MinimumInterval : IntervalFrequency : MaximumInterval));
>     IntervalFitnessP = zeros( [ kMax 1 ] );
>     IntervalGapP     = zeros( [ kMax 1 ] );
>     parfor k = 1:kMax
>     %for i = MinimumInterval : IntervalFrequency : MaximumInterval
>         i = (k - 1) * IntervalFrequency + MinimumInterval;
1307,1308c1315,1318
<         IntervalFitness( (i + 1) - MinimumInterval ) = max( GapsConfidence );
<         IntervalGap( (i+1) - MinimumInterval )       = GapPeaks( 1 );
---
>         %IntervalFitness( (i + 1) - MinimumInterval ) = max( GapsConfidence );
>         %IntervalGap( (i+1) - MinimumInterval )       = GapPeaks( 1 );
>         IntervalFitnessP(k) = max(GapsConfidence);
>         IntervalGapP(k) = GapPeaks(1);
```

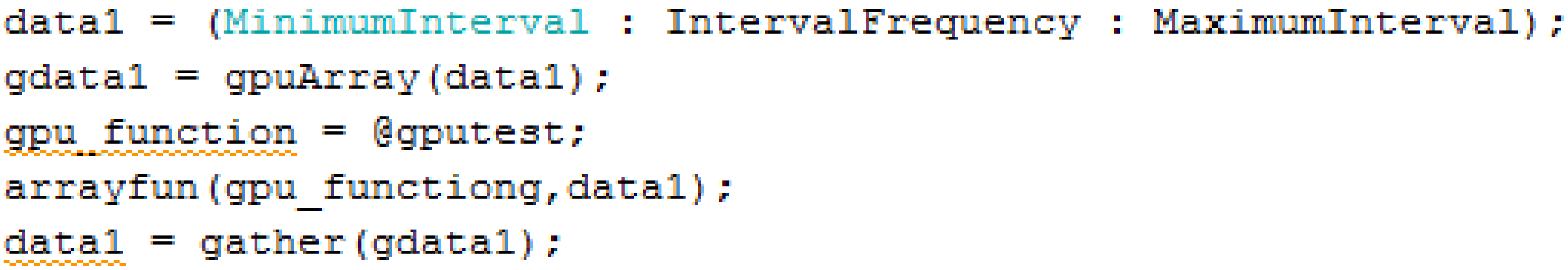
CPU Parallelization – Results

	base	parfor	%
timeTest	126.4	47.2	37.5%
timeFit	59.3	30.4	51.3%
timeBpm	186.0	77.7	41.8%
timeProgram	**202.7**	**93.3**	**46.0%**



GPU Parallelization – Approach

- MATLAB’s gpuArray() and gather() function
- MATLAB’s build-in GPU functions
- Parallel GPU kernel by using arrayfun()

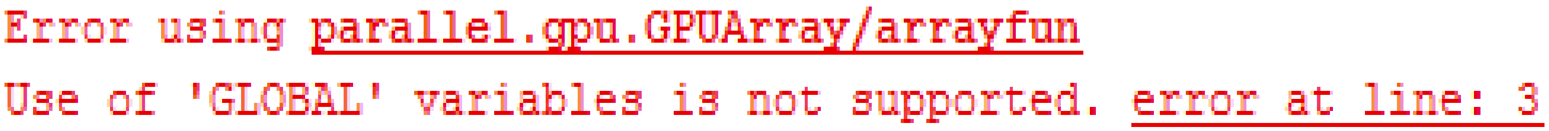


```
data1 = (MinimumInterval : IntervalFrequency : MaximumInterval);
gdata1 = gpuArray(data1);
gpu_function = @gputest;
arrayfun(gpu_functiong,data1);
data1 = gather(gdata1);
```

http://www.mathworks.com/help/toolbox/distcomp/bsic3by.html

GPU Parallelization – Issues

- Global variables/data structures

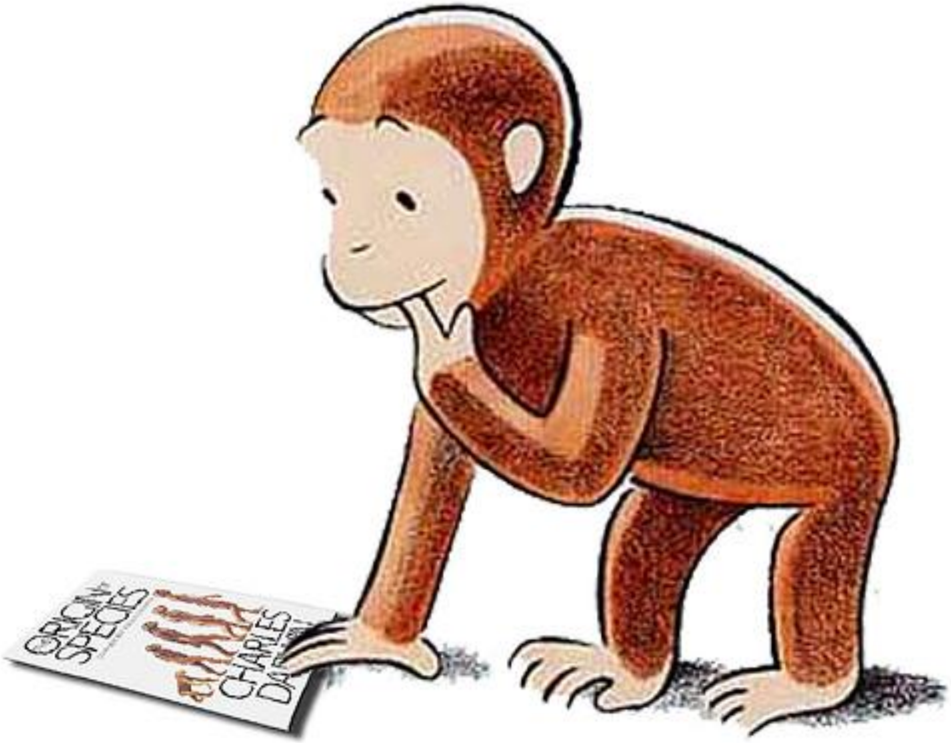


```
Error using parallel.gpu.GPUArray/arrayfun
Use of 'GLOBAL' variables is not supported. error at line: 3
```

- Rewrite code
  - Loops -> GPU Kernel functions
  - Data -> eliminate their cohesion and modify their type so that they can be used in GPU Kernel
- Slow memory copy

	base	With data transform	%
timeProgram	26.6	49.2	185.0%

Questions?



- Blog: http://dancingmonkeysaccelerated.blogspot.com/
- Code: https://github.com/Keripo/DancingMonkeysAccelerated

img src: http://www.gratuitousscience.com/wp-content/uploads/2010/04/6a00d83451f25369e200e54f94996e8834-800wi.jpg