Artificial Intelligence and Java

Zoran Sevarac,
University of Belgrade, Faculty of Organisational Sciences
sevarac@gmail.com
@zsevarac

jCrete2016

What is this all about?

- Ai is the **big buzz** at the moment
- Machine learning, and deep learning are pushing forward
- Almost all new developements are on other platforms (C++, Python)
- Existing Java solutions are slower, more complex to use, and often rely on native libraries (portability issues)
- Java as a development platform for AI is falling behind
- What can be done about that?

Hot Buzzwords – What is out there?

- Machine Learning
- Deep Learning / Neural Networks
- Big Data
- Analytics
- Prediction
- Cognitive Applications/Systems
- Computer vision / Image recognition
- Speech Recognition
- Natural Language Understanding
- Cloud Ai
- Ai Assisstants

AI is the next big thing,

and we have to make

Java no. 1 development platform

for it now!

Current Situation (for DeepLearning)

- Most deep learning frameworks implemented in C++, Python or Lua (Caffe, Torch, Tensor Flow)
- GPUs are currently the horse power of the AI
- Using Cuda or OpenCL for parallel computing on GPU
- Java falling behind in performance, too complex to develop and use, problems with portability

Possible Solutions

- Using GPU-s From Java (jcuda, jocl – also use native libs, aparapi – slow, LWJGL?)

- Using native libs from Java (jblas/OpenBLAS)

Problems with those

- Using native libraries brings problems with portability, performance and usability
- Native calls are expensive:
  - Arrays are always copied on native calls.
  - For large arrays lots of cache misses
- GPU limited in memory

It would be so nice to have

- Vector operations runing on multicore CPU, with speed close to GPU
- Faster and easier integration with native Libraries and GPU
- Easy programming model (no OpenCL, Cuda code)
- Support for fast parallelized matrix operations in pure Java

Current Attempts That Might Help in future

- **Project Panama** (improving native calls, support for vector operations?)

  http://openjdk.java.net/projects/panama/
- **Project Sumatra** (use GPUs and APUs from Java)

  http://openjdk.java.net/projects/sumatra/
- **Vector API?**
- **Graal** (compiler that can customize generated gode and replace parts of byte code with vector CPU or GPU instructions)

  http://openjdk.java.net/projects/graal/

Code & Benchmarking

- Basic Matrix Multiplication
- Basic Matrix class, stores all data in a single array in order to easily support N-dimensional arrays, and its more efficient (Sven said that!:)
- Basic example, making sure that get and set methods are inlined

Benchmark Using JMH

- Single and multi threaded matrix multiplication
- Squared matrix of double values
- Sizes: 100x100, 500x500, 1000x1000, 2000x2000, 3000x3000
- O(n) = n^3
- 5 Warm-Up Iterations
- 5 Measurement Iterations
- Java 1.8
- Intel® Core™ i7-4500U CPU @ 1.80GHz × 4 , 16 GB RAM, Ubuntu

JMH Benchmark Results

Benchmark mode: Throughput
Measures the number of times per second benchmark method could be executed.

Matrix Dimension	Single Thread (ops/s)	Two Threads (ops/s)	Speed-up
100x100	914.224	825.448	0.902
500x500	6.508	16.721	2.569
1000x1000	0.491	1.238	2.521
2000x2000	0.039	0.105	2.69
3000x3000	0.014	0.031	2.21

Conclusion

- Matrix operations on Java could be at least 2 times faster using multithreading, when matrices are large enough (more then 2500 elements).
- On small matrices multi threading is slower
- This approach provides significant speed-up without using native libraries or GPU
- Would adding more threads make things even faster? - Very likely
- Can pure Java on CPU with specialised implementations beat GPU? Work in progress...
- If that could be achieved, it would be big win for Java in the area of Ai (big data, machine learning and scientific computing)

Confidential Hidden Slide That Only You Can See

- I'm coming from future, year 2032. to warn you that you must improve JVM performance and portability for Artificial Intelligence applications. Otherwise our robot army will loose the determining battle against enemies of human kind, and we willl forever stay enslaved.
- The improvements must be released with Java 9, in order to secure the chain of events that will lead to the victory of the human kind.
- So... tweet about it to save the world!