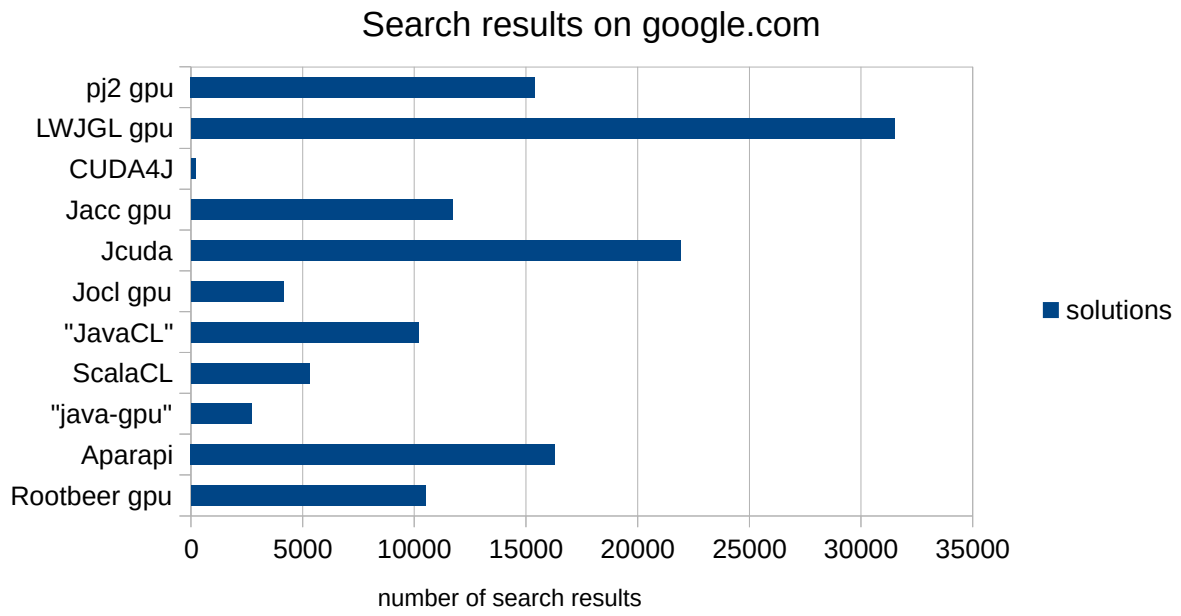
Liste des candidats[1]

Nom	Description	Lien
(Byte)code translation and OpenCL code generation		
aparapi	An open-source library that is created and actively maintained by AMD. In a special "Kernel" class, one can override a specific method which should be executed in parallel. The byte code of this method is loaded at runtime using an own bytecode reader. The code is translated into OpenCL code, which is then compiled using the OpenCL compiler. The result can then be executed on the OpenCL device, which may be a GPU or a CPU. If the compilation into OpenCL is not possible (or no OpenCL is available), the code will still be executed in parallel, using a Thread Pool.	https://github.com/aparapi/aparapi
rootbeer1	An open-source library for converting parts of Java into CUDA programs. It offers dedicated interfaces that may be implemented to indicate that a certain class should be executed on the GPU. In contrast to Aparapi, it tries to automatically serialize the "relevant" data (that is, the complete relevant part of the object graph!) into a representation that is suitable for the GPU.	https://github.com/pcpratts/rootbeer1
java-gpu	A library for translating annotated Java code (with some limitations) into CUDA code, which is then compiled into a library that executes the code on the GPU. The Library was developed in the context of a PhD thesis, which contains profound background information about the translation process.	http://code.google.com/p/java-gpu/
ScalaCL	Scala bindings for OpenCL. Allows special Scala collections to be processed in parallel with OpenCL. The functions that are called on the elements of the collections can be usual Scala functions (with some limitations) which are then translated into OpenCL kernels.	https://github.com/ochafik/ScalaCL
Language extensionsJava OpenCL/CUDA binding libraries		
JavaCL	A language extension for Java that allows parallel constructs (e.g. parallel for loops, OpenMP style) which are then executed on the GPU with OpenCL. Unfortunately, this very promising project is no longer maintained.Java bindings for OpenCL: An object-oriented OpenCL library, based on auto-generated low-level bindings.	https://github.com/ochafik/JavaCL
jocl	Java bindings for OpenCL: An object-oriented OpenCL library, based on auto-generated low-level bindings.	http://jogamp.org/jocl/www/
lwjgl	Java bindings for OpenCL: Auto-generated low-level bindings and object-oriented convenience classes.	http://www.lwjgl.org/
jocl	Java bindings for OpenCL: Low-level bindings that are a 1:1 mapping of the original OpenCL API.	http://jocl.org/
jcuda	Java bindings for CUDA: Low-level bindings that are a 1:1 mapping of the original CUDA API.	http://jcuda.org/
Jacc	Java frameworks that performs on low-level at the runtime. Experimental phase.	http://arxiv.org/pdf/1508.06791.pdf
CUDA4J	Java API made by IBM that gives many classes for GPU computing via CUDA peripheral.	https://www-01.ibm.com/support/knowledgecenter/SSYKE2_7.0.0/com.ibm

1 Partially taken from http://stackoverflow.com/questions/22866901/using-java-with-nvidia-gpus-cuda/22868938#22868938 – edited Mar 17'15

Nom	Description	Lien
		.java.lnx.71.doc/user/gpu_developing_cuda4j.html
Com.ibm.gpu	Another java api made by ibm to sort primitive arrays via gpu.	https://www-01.ibm.com/support/knowledgecenter/SSYKE2_7.0.0/com.ibm.java.lnx.71.doc/user/gpu_developing_sort.html
lwjgl	Lightweightjava game library. A game library that gives the ability to use OpenCL for parallel programming.	https://www.lwjgl.org/
PJ2	Parallel Java 2 (PJ2) is an API and middleware for parallel programming in 100% Java on multicore parallel computers, cluster parallel computers, hybrid multicore cluster parallel computers, and GPU accelerated parallel computers[2].	https://www.cs.rit.edu/~ark/pj2.shtml



2 https://www.cs.rit.edu/~ark/pj2.shtml