Increase ImageJ plugin performance by using JNI (Java Native Interface)

Abstract:
JavaTM Native Interface (JNI) is a standard programming interface for writing Java native methods and embedding the Java virtual machine into native applications.

It allows Java code that runs inside a Java Virtual Machine (VM) to inter-operate with applications and libraries written in other programming languages, such as C, C++, and assembly.

The most important benefit of the JNI is that it imposes no restrictions on the implementation of the underlying Java VM. Therefore, Java VM vendors can add support for the JNI without affecting other parts of the VM. Programmers can write one version of a native application or library and expect it to work with all Java VMs supporting the JNI.

JNI can be found extremely useful to use well known imagery treatment libraries written in C++ into ImageJ plugins (such as FFTW library to perform fast N dimensions Fourier Transform), and also to fasten up the intensive CPU processing parts of the Plugin algorithm.

In this paper we shall discuss the methods and the IDEs (Integrated Development Environment) to be used for a proper JNI implementation, the Consequences of such implementation, and also the adequate insertion of the resulting shared libraries into ImageJ environment that respect the “Plug and Play” concept of a plugin.

We finish by a brief presentation of Plugins using this technique, developed in our laboratories for 3D fluorescence microscopy imagery treatment. (PSF simulation, Fourier Transform, iterative and direct Deconvolution …etc).

Keywords:
Java Native Interface, performance, ImageJ Plugins

Author
Elie Maalouf, Bruno Colicchio and Alain Dieterlen

Organisation
University of Mulhouse Colmar

Homepage http://www.uha.fr

Short Biography