# Evaluating the utility of Java Native Interface (JNI) in HPC with Java

Sameer Puri

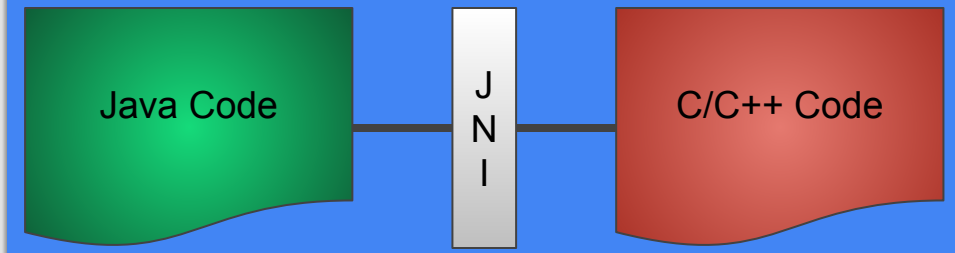
# Intro

- Popular OOP HPC languages: C++ and Java
- Higher performance ceiling for C++
  - JVM and bytecode in Java
  - Native system-specific binaries in C++
- Faster development with Java
  - “Core dumped” / “segfault” in C++
  - Stack traces & system independence in Java

# JNI

Allows native code in Java

“The performance gap bridger”



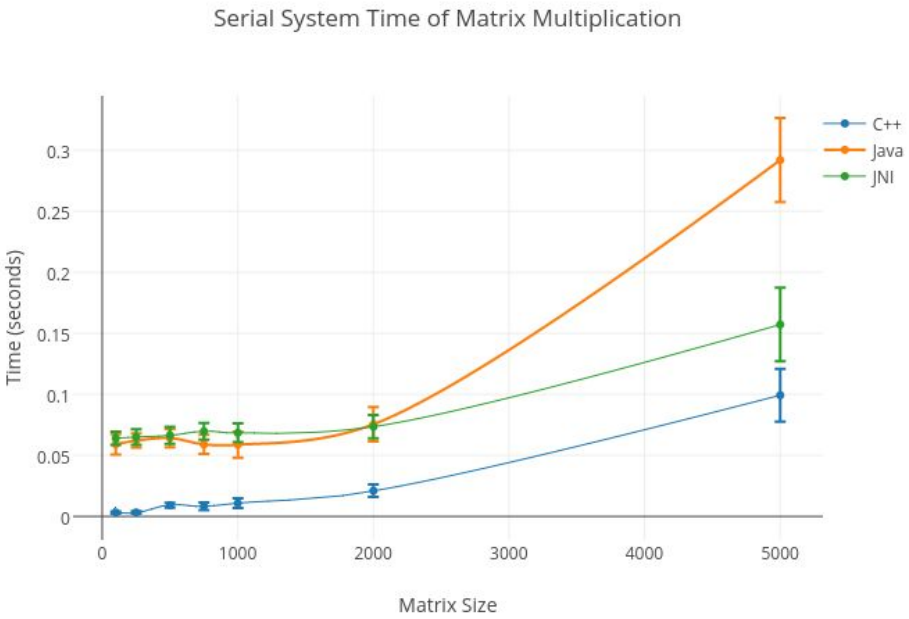
# BQ: Is JNI performant enough to justify using Java in HPC?

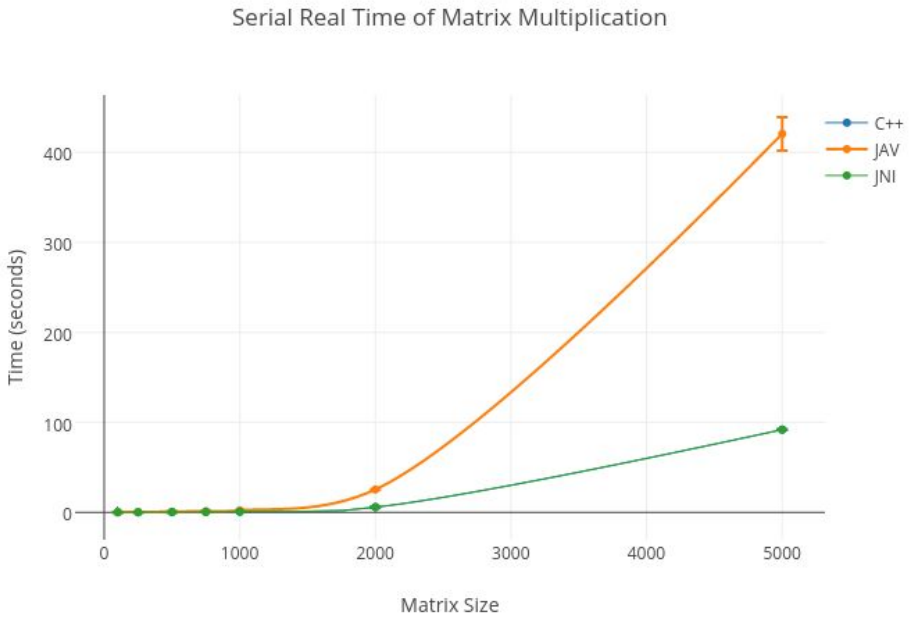
Compilers:
javac 1.8.0_45, icpc++ 14.0.2

# Matrix Multiplication

- Generate n x n matrix
  - All $a_{ij}$ in $A_{n \times n} = i * j$
- Multiply by self
- Print reduce operation
  - Add elements together and print

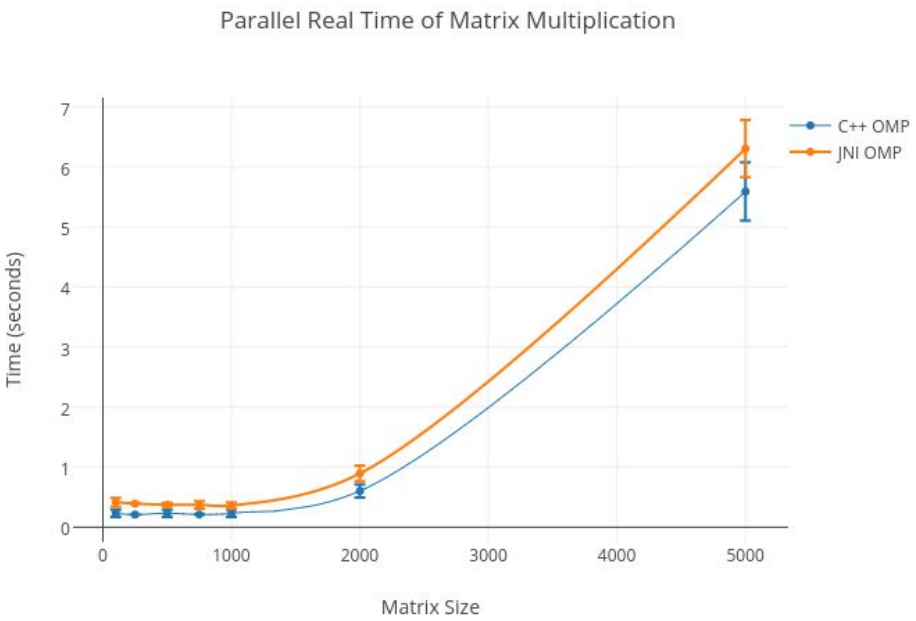
- Types: C++, Java, JNI, JNIOMP, C++OMP
- Sizes: 100, 250, 500, 750, 1000, 2000, 5000
- 11 Trials
- Output: real, user, sys time

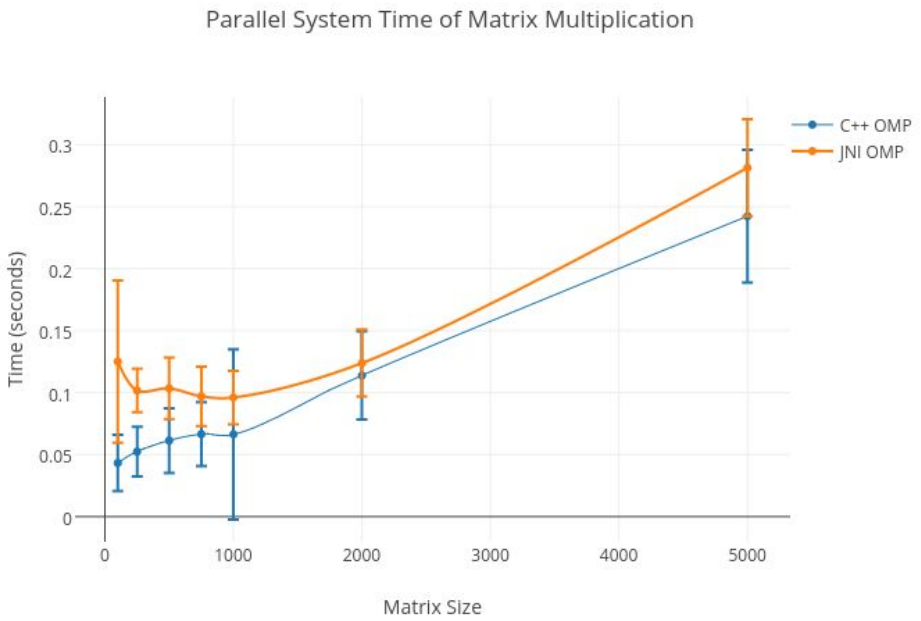



# Serial Time Results

- Java performs poorly
- JNI performs almost the same as C++ : consistent system overhead
- Java system overhead grows



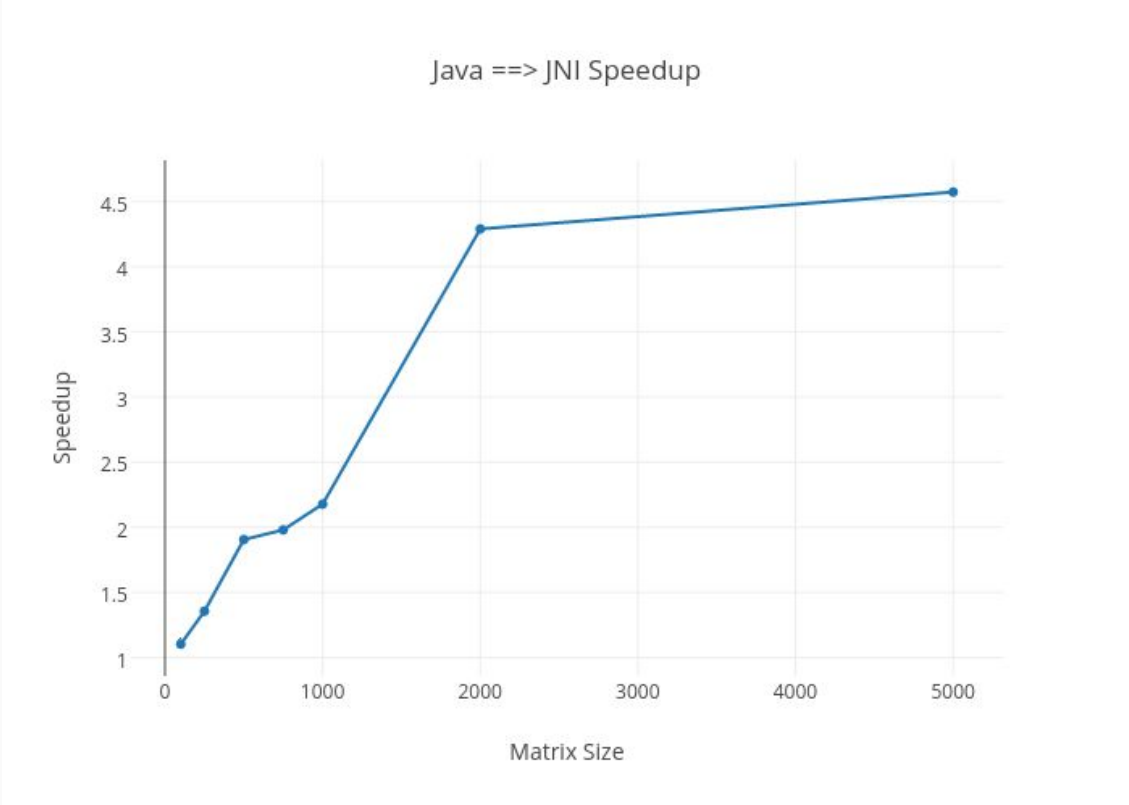




# Parallel Time Results

- JNI has consistent real time overhead
- System time is fairly variable

# Takeaways

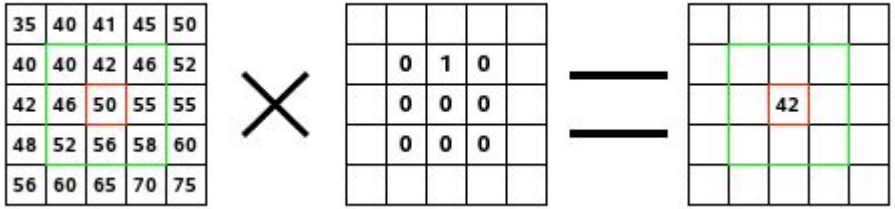
- JNI provides performance benefits of native code
- Constant overhead in JNI
- Per loop iteration overhead in Java
- JNI has serious advantages →



# Matrix Convolution Program in Java

- Element-by-element transformation operation on a matrix using a 3x3 “kernel” matrix
- Used in Image Filters
  - Sharpen
  - Posterize
  - Edge Trace

- Types: Java, JNI, JNIOMP
- 11 Trials
- Output: time of matrix_convolution function
- Ignored full time -- large file I/O causes lots of variance




Source: https://docs.gimp.org/en/plug-in-convmatrix.html

VANDERBILT UNIVERSITY

ACCRE

Advanced Computing Center for Research & Education

*Enabling Researcher-Driven Innovation and Exploration*

VANDERBILT UNIVERSITY

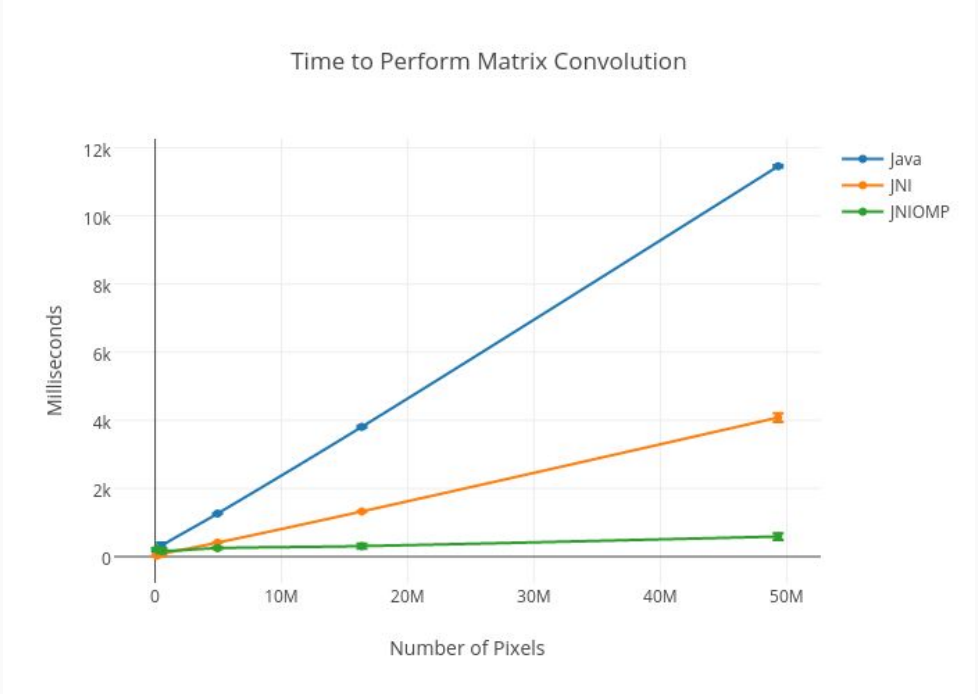
ACCRE

Advanced Computing Center for Research & Education

*Enabling Researcher-Driven Innovation and Exploration*

VANDERBILT

VANDERBILT




# Time Results

- JNI slices off a big portion of overhead
- OMP does even more!

# JNI makes using Java in HPC viable

Questions?