

Java Grande Introduction

- Grande Application: a GA is any application, scientific or industrial, that requires a large number of computing resources(CPUs, networks, I/O, memory)
- Java Grande Forum:
 - Initiated by some people in academic
 - Supported by IBM, SUN, NIST ...
- Primary Goal: Making Java a better environment for Grande Application development

Advantages/ Disadvantages of Java

- Supporters of Java Grande believe:
 - Java can be written once and run virtually everywhere: Ideal for heterogeneous computing
 - Better floating point computing predictability
 - ...
- However, Java is not perfect
 - Numeric computing performance is not very satisfying
 - The performance should be comparable to what is achieved in C/Fortran if it is a choice for high-performance computing
 - There are spaces for improvement on memory management and concurrency performance.

Improving Numeric Computation Performance

- Java needs to be improved to be a suitable environment for high performance numeric computation
- Constraints of the changes:
 - Relatively small change on JVM
 - Upward compatibility and a big amount of backward compatibility should be maintained
 - Good execution speed should be achieved on widely available microprocessors

Critical Issues

- Lightweight classes
 - Implementation of alternative arithmetic such as complex, multiple precision and so on requires new objects with value semantics
- Operator overloading
 - Alternative arithmetic operation should be as readable as those based only on float and double
- Multidimensional Arrays
 - Operations on multidimensional arrays of elementary numerical types must be easily optimized and the lay out of arrays should be known to the developer

Complex Arithmetic

- Straight forward complex class is far from efficient:
 - Object overhead
 - Semantics of complex objects are different from those of float/double
 - Use method calls for arithmetic operations leads to very bad code
- Add a new primitive type
 - A lot of changes have to be done on JVM or
 - Mapping the complex type into a pair of double
 - How to pass complex into a method
 - How to return a complex number
 - And what about other numeric types?

The answer: using light weighted classes



Light weight classes

- Allow runtime use of what appears to be a C struct
 - Having value semantics
 - Never be null
 - No dynamic dispatch overhead is needed since these classes are always final
- Language modification
 - A new class modifier
- JVM level
 - Add a new struct (big change to JVM), or
 - Translate lightweight classes in to normal classes by the compiler. The compiler should also be responsible for the optimization of space and speed (good backward compatibility)

Operator Overloading

- Without overloading, codes using complex arithmetic would look very different than code using real arithmetic
- A minimum set of existing operators need be overloaded
 - Arithmetic, comparison, assignment
- Problems:
 - How to name the methods corresponding to overloaded operators: + are not allowed in Java to form an identifier
 - How to resolve overloaded methods:
 - Double + Complex
 - Complex + Double

Multidimensional Array

- Multidimensional arrays are very useful in scientific computation
- Java's array of array is not real multidimensional array
 - `Double [][] A = new double[m][n];`
 - `A[i] = A[i] + 1;`
- Suggestion:
 - For each primitive type and some other types, and each rank (0 –7) , build a standard java class and put it into the library
 - Allow the concept of regular section: subarray of another array
 - Expose the internal storage scheme to the developer

To be continued (if there is a chance)

- Java in scientific computation and enterprise computation
- How to improve concurrency performance
- How to manage memory more efficiently