Multiple Dispatch in the Java Virtual Machine

What is (dynamic) Multiple Dispatch?
Method selection based on the types of
... all (or more than one) of the arguments
... at execution time (not compile time).

Isn’t this Method Overloading?
The Container Problem shows that static multiple dispatch
(method overloading) alone isn’t enough.

But, how common is this problem?
This occurs frequently enough that a solution (Visitor) was included
the original GOF design patterns; common situations include:

Event Programming
Drag and Drop
Binary Operations

Previously, how did people work around this problem?

Visitor Pattern
Type Testing
Type Numbering

How does the Multiple Dispatch version look like?

Native multiple dispatch in the Java Virtual Machine

How efficient is it?
High-performance table-based multiple dispatcher operates faster than
double-dispatching in the Java interpreter.

Multiple Dispatch AWT and SWING
We modified SWING/AWT to use multiple dispatch.
✓ modified 92 out of 846 classes
✓ removed 123 custom-coded dispatchers
✓ eliminated 171 decision points (if/else and switch/case statements)
✓ reduced the average decision points per method from 3.8 to 2.0
✓ replaced 4.74 million dispatches with 2.35 million multi-dispatches
... with no measurable performance difference

What are the benefits of your approach?
• No language extensions:
✓ keep existing tools: debuggers, profilers
✓ source-code and binary compatible
✓ 3rd party libraries extensible to multimethods without source
• Programmer-targeted multiple dispatch – single dispatch unchanged
• Full Java support: invokevirtual, invokespecial