An Exploration of Specialized Java Bytecodes:

Do Specialized Bytecodes Have Value?

Ben Stephenson

University of Western Ontario
Outline

- Background
- Motivation, Purpose and Hypothesis
- Despecializations
- Specializations
- Performance Results
- Conclusion
Java Bytecodes are a representation used to express a Java program

- Core Bytecodes: provide unique functionality
- Specialized Bytecodes:
  - Functionality for some bytecodes is duplicated exactly by another bytecode
  - Some others can easily be mimicked with a short sequence of other bytecodes
Motivation

- JVM Specification defines 201 bytecodes
  - “Leftover” bytecodes are already being used for implementation specific functionality
- It is desirable to create new Java bytecodes to
  - Support new language features efficiently
  - Support for new native data types
  - Introduce new bytecodes that allow common operations to be performed more efficiently
Purpose and Hypothesis

• **Purpose:** To determine the runtime performance impact of specialized bytecodes

• **Hypothesis:** Specialized bytecodes offer a performance benefit
Despecializations

• Despecializations remove specialized bytecodes by expressing their functionality with equivalent core bytecodes
  – Consider the bytecode iload_1
    • Loads variable at index 1 onto the stack as an integer
  – Functionality is duplicated by the iload bytecode
    • Loads an arbitrary variable onto the stack as an integer

• 67 possible despecializations have been identified
Despecializations

<table>
<thead>
<tr>
<th>Before:</th>
<th>After:</th>
</tr>
</thead>
<tbody>
<tr>
<td>n-1</td>
<td>n-1</td>
</tr>
<tr>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>n+1</td>
<td>n+1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>iload_1</td>
<td>iload</td>
</tr>
<tr>
<td>bytecode</td>
<td>1</td>
</tr>
<tr>
<td>bytecode</td>
<td>bytecode</td>
</tr>
</tbody>
</table>
Specializations

• Introduce a new bytecode to replace instances of a core bytecode with a specific argument
  – Example: iload 05 becomes iload_5

• Some core bytecodes with a specific argument are more common than current specialized bytecodes
Profiling Results

Frequency of Execution of Core and Specialized Bytecodes

Percentage of Total Bytecodes Executed

- Specialized Bytecode
- Core Bytecode

10.19494636
Performance Test Conditions

- Despecialized all specialized bytecodes that occur less than 0.1 percent of the time (28 despecializations)

- Specialized all core bytecodes with specific arguments that occur at least 0.1 percent of the time (42 specialization)

- Switched infrequently used specialized bytecodes with frequently occurring core bytecodes (31 switches)
Performance Results

• Most benchmarks showed small differences
• Average difference across all benchmarks was less than 1.0 percent for all 3 conditions
• Repeated Measures Analysis of Variance concluded that differences are not statistically significant at $p$ less than 0.05
• Original hypothesis does not appear to hold
Conclusions

• Existing specialized bytecodes have little impact on application runtime
• Picking more frequently occurring specialized bytecodes does not improve application performance
• Careful consideration should be given
  – to removing specialized bytecodes from JVM
  – by anyone developing a new VM
Summary

• Java bytecodes can be categorized as “core” or “specialized”
  – Specialized bytecodes can be replaced with core bytecodes
  – New specialized bytecodes can be introduced

• Performance results showed no meaningful difference in performance

• Value of specialized bytecodes should be carefully considered
Questions?