Evaluation of the runtime performance of control flow structures for dynamic dispatch in Java

Olivier Zendra, Karel Driesen, Feng Qian, Laurie Hendren

Evaluation of the runtime performance of control flow structures for dynamic dispatch in Java

Olivier Zendra & Karel Driesen (INRIA − McGill University, ACL group) − Feng Qian & Laurie Hendren (McGill University, Sable group)
{ozendra,karel,fqian,hendren}@cs.mcgill.ca

Motivation:

Frequent dynamic dispatch in Java programs

➪ Efficient implementation crucial

➪ Which bytecode control structure is the most efficient to implement dynamic dispatch ?

Methodology:

A benchmark suite to time control structures for dynamic dispatch in Java under varying conditions:
- Different control structures
- Different numbers of possible receiver types (static)
- Different patterns of runtime receiver types (dynamic)
- Different execution environments

Java benchmarks are generated for each combination of control structure and number of possible types
- ➪ 220+ variants

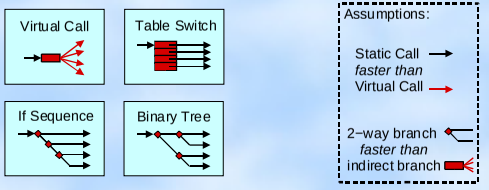
Type ID pattern files are generated, representing the order in which types occur at runtime
- ➪ Control & repeatability
- ➪ 60+ variants

We time the execution of each benchmark on all patterns that apply to it, on various execution environments:
- Java Virtual Machines:
  - −SUN HotSpot Server [HSS]
  - −SUN HotSpot Client [HSC]
  - −IBM JIT
- Hardware platforms:
  - −Intel Pentium III [P3]
  - −SUN UltraSparc III [U3]
  - −(Intel Celeron)
  - −(AMD Athlon)

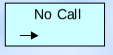
➪ Bench × pattern × JVM × platform ≃ 21,000 data points

Control structures:

We generate different control structures for polymorphic calls:



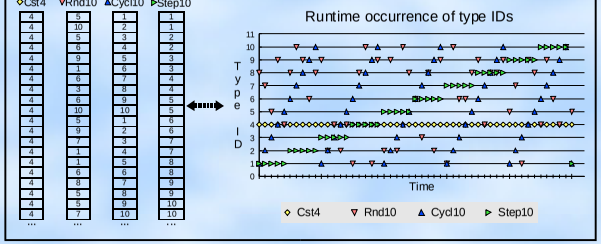
For comparison we also run benchmarks without any call:



Type ID Patterns:

	◇ Constant	▽ Random	△ Cyclic	▷ Stepped
Polymorphism	None	High	High	High
Variability	None	High	High	Low
Predictability	100%	None	Average	High
Frequency	Very common	Uncommon	Uncommon	Common

Implementation: x,000 iterations over a 10Kint array −> x0,000,000 calls



Some results:

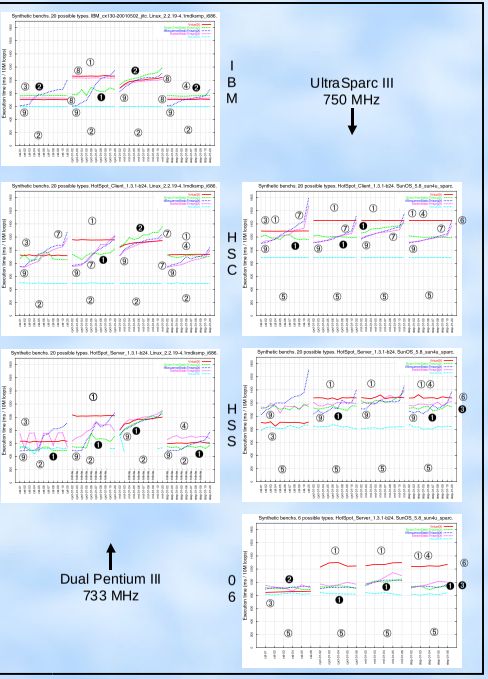
① Virtual calls tend to be more expensive than other structures
② Virtual calls on P3 are sensitive to execution patterns
③ Virtual calls on constant patterns tend to be efficient on P3 and U3
④ Virtual calls with stepped patterns behave well on P3 but not on U3
⑤ Virtual calls on U3 are insensitive to execution patterns, except for the constant pattern
⑥ Virtual calls on U3 tend to be costly

⑦ Switches on HSC behave like if sequences
⑧ Switches on IBM behave like virtual calls

⑨ If sequences are the best when only the first few ifs are exercised, except in some cases for HSS on U3

❶ Binary trees are significantly faster than virtual calls in most cases
❷ When binary trees are slower than virtual calls, the difference generally remains small

❸ All control structures, except virtual calls, speed up when the number of possible types is small



Conclusion:

- Performance of dynamic dispatch varies greatly across VMs and execution patterns
- Dynamic dispatch in current VMs can be significantly improved at compile time
- Much better implementations are possible for small numbers of possible receiver types: if sequences for up to ≃4, binary tree dispatch up to ≃8