Inline Caching meets Quickening

Stefan Brunthaler

Institut für Computersprachen
Technische Universität Wien

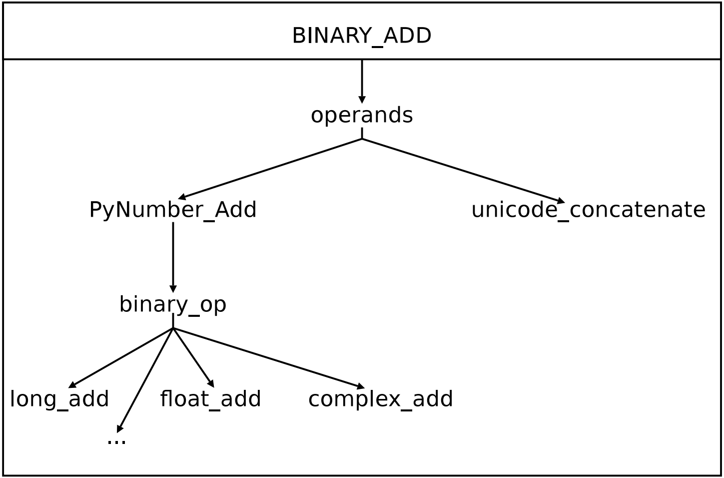
Smalltalk

Deutsch and Schiffman: “Efficient Implementation of the Smalltalk-80 System”, 1984:

“dynamic locality of type usage”

The actual operand types for an instruction within a sequence tend to remain fixed for about 95% of the time.

Inline Caching



Ad-hoc polymorphism in Python 3

No Inline Caching

Many popular interpreters do *not* use inline caching.

- Perl
- Python
- Ruby
- Tcl
- ...

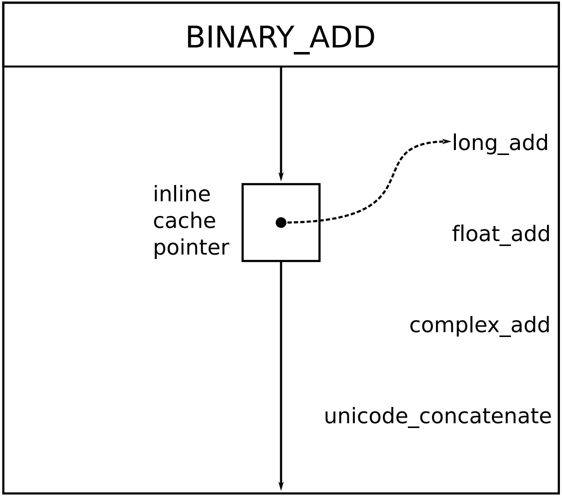
None of the above have dynamic translators, or use known hash-table based look-up caches as used in Smalltalk implementations of the early 80s.

Why do we care?

1000 : 10 : 1

Ertl, M. A. and D. Gregg, *The structure and performance of efficient interpreters.*, J. Instruction-Level Parallelism **5** (2003).

Adding Inline Cache Pointer



`BINARY_ADD` *instance* with `long_add` cached.

Adding Inline Cache Pointer

Efficient basic technique without dynamic translation:

- transform into regular instruction format
- interleave instructions and inline cache pointers
- relocate jump offsets

Discussion

Pros:

- easy to implement
- more efficient than Smalltalk-style look-up caches (**no** hash-tables)
- improved data locality

Cons:

- memory consumption ➟ profiling
- additional indirect call

What is quickening?

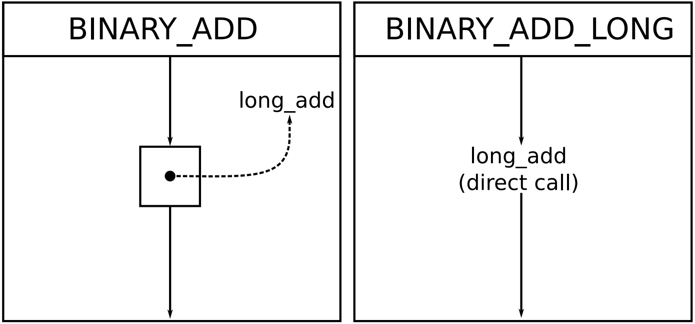
The Java virtual machine uses *quick instructions*, i.e., some instructions are replaced by more efficient *quick* instructions after the first execution. The non-quick instruction usually does initialization work.

Quickening specializes instructions with respect to their operand values.

Idea

Create optimized derivatives based on result of **resolving ad-hoc polymorphism**.

Inline Caching meets Quickening



Replacing the indirect call with a direct call.

Discussion

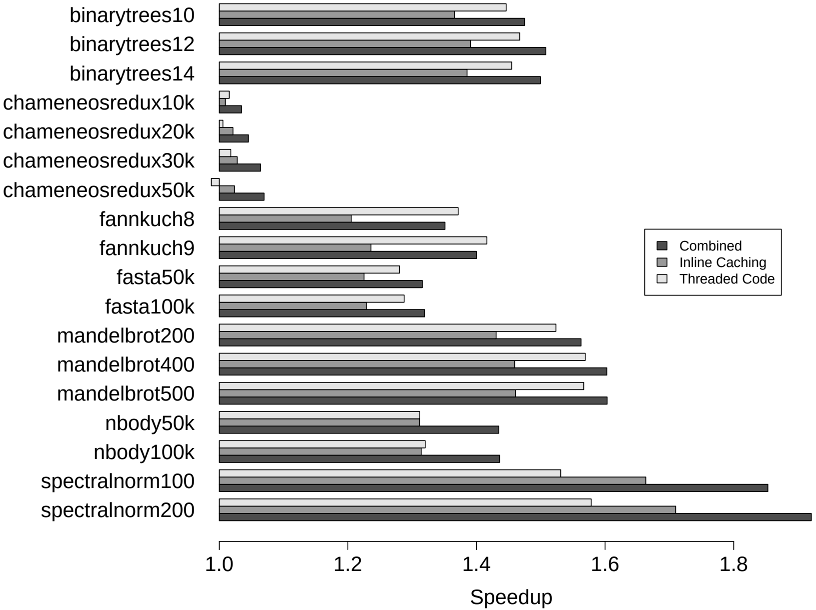
Pros:

- easy to implement
- more efficient than previous technique
  ➡ eliminates indirect call
- enables inlining of functions by compiler
- does not require changing instruction format (for many use cases)

Cons:

- requires instruction set extension ➡ generator
- increased instruction cache requirements
  ➡ desktop/server vs. mobile devices

Benchmarks


binarytrees10
binarytrees12
binarytrees14
chameneosredux10k
chameneosredux20k
chameneosredux30k
chameneosredux50k
fannkuch8
fannkuch9
fasta50k
fasta100k
mandelbrot200
mandelbrot400
mandelbrot500
nbody50k
nbody100k
spectralnorm100
spectralnorm200
Combined
Inline Caching
Threaded Code
1.0
1.2
1.4
1.6
1.8
Speedup

Take away message

Efficient inline caching is possible *without* dynamic translation—and *worthwhile*.