

CastOff: A Compiler for Ruby1.9.3

Satoshi Shiba

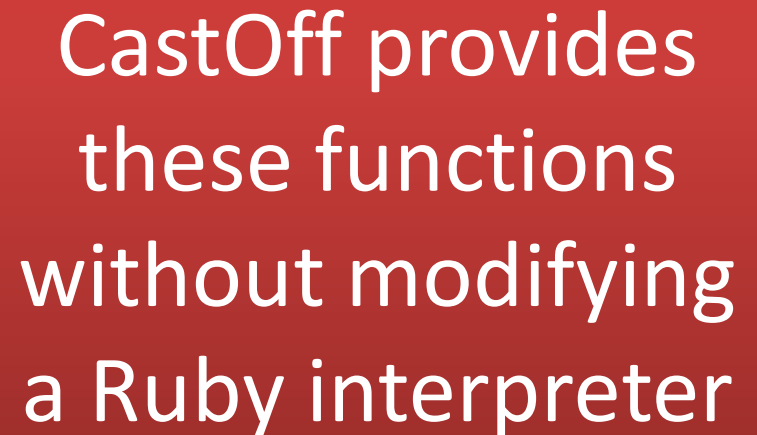
OUTLINE OF CASTOFF

Outline of CastOff

- **CastOff: A Compiler for Ruby Implemented as a Library**

- Functions:

- Runtime compilation
- Profiling execution
- Deoptimization
- Re-compilation
- Annotation support
- Reuse of compiled codes



CastOff provides
these functions
without modifying
a Ruby interpreter

- CastOff is hosted on [Rubygems.org](https://rubygems.org)

- Installation: **gem install cast_off**
- Command line tool `cast_off` is available after installation

Behavior of CastOff [1/3]

```
# Command line  
$ cast_off fact.rb 10
```

Behavior of CastOff

- Run and profile “ruby fact.rb 10”
- Detects *fact* as “hot” method
- Expects following condition
 - **variable *i* in *fact* is Fixnum obj**
 - ***fact* returns Fixnum obj**
 - ...
- Compile *fact*

```
# fact.rb  
def fact(i)  
  i > 1 ? (i * fact(i - 1)) : 1  
end  
fact(ARGV.shift.to_i)
```

sample script

- Profiling execution

Behavior of CastOff [2/3]

```
# Command line  
$ cast_off -run fact.rb 10
```

Behavior of CastOff

- Load and execute fact.rb
 - Detects definition of *fact*
 - **Replace original *fact* to compiled *fact***
- Run compiled *fact*

```
# fact.rb  
def fact(i)  
  i > 1 ? (i * fact(i - 1)) : 1  
end  
fact(ARGV.shift.to_i)
```

sample script

- Reuse of compiled codes

Behavior of CastOff [3/3]

```
# Command line  
$ cast_off -run fact.rb 100
```

Behavior of CastOff

- Load and execute fact.rb
 - Detects definition of *fact*
 - Replace original *fact* to compiled *fact*
 - Run compiled *fact*
- **Detects Bignum obj**
- **Deoptimize compiled *fact***
- **Re-compile fact and load**

```
# fact.rb  
def fact(i)  
  i > 1 ? (i * fact(i - 1)) : 1  
end  
fact(ARGV.shift.to_i)
```

sample script

- Deoptimization
- Re-compilation
- Runtime compilation

Additional function

- User can specify information to CastOff directly
 - Compilation target and timing
 - Type information of variables, method return values
 - CastOff can combine annotation and profiling results



Programmer

```
# user annotation
CastOff.compile_singleton
on_method(
  self, :fact,
  :i => [Fixnum]
)
```

```
def fact(i)
  i > 1 ? (i * fact(i - 1)) : 1
end
```

```
CastOff.compile_singleton_
method(...
```

```
fact(ARGV.shift.to_i)
```

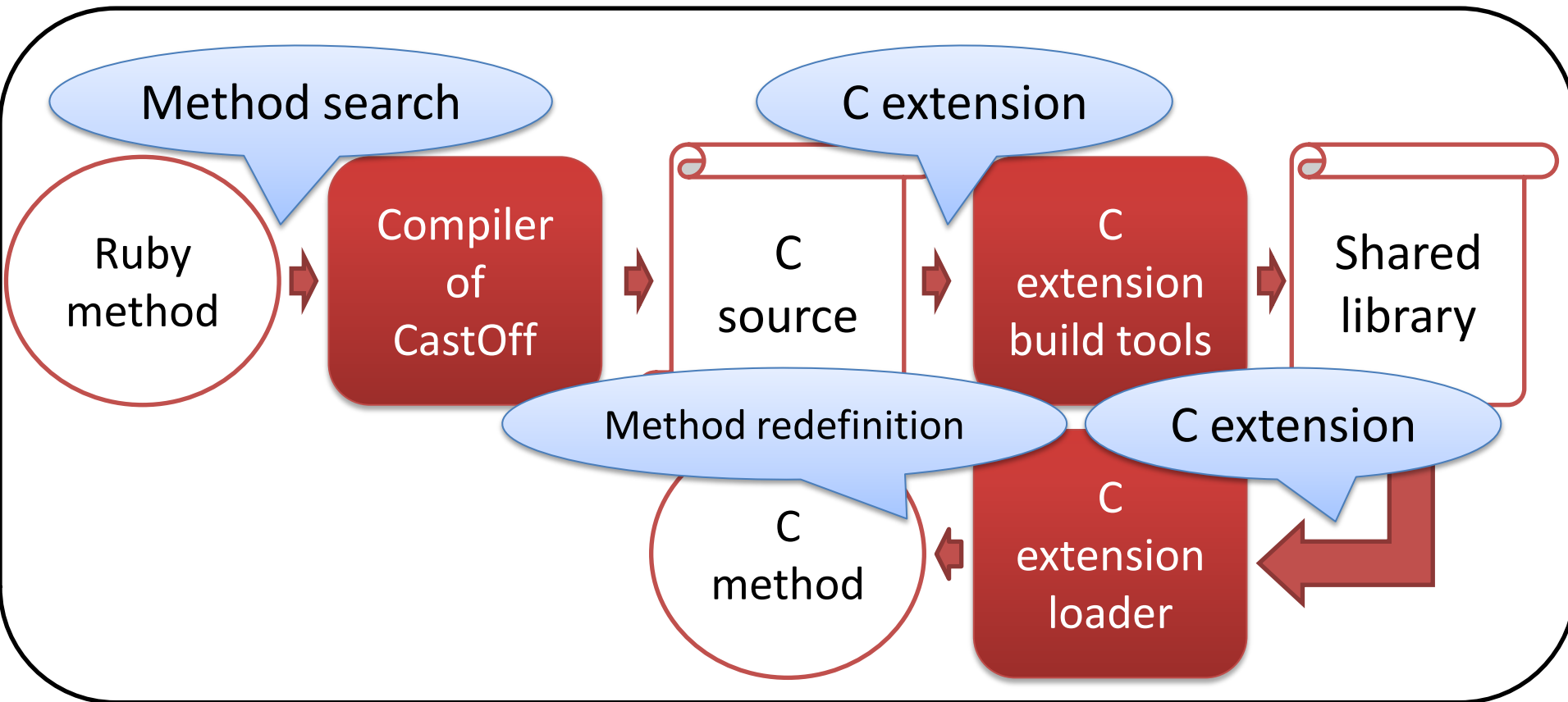
Optimization of CastOff

- Current optimization
 - Devirtualization
 - Redundant String literal duplication elimination
 - Block inlining
 - Unboxing
 - Constant prefetch
- Future optimization
 - Classical optimizations
 - ⇒ Method inlining, Constant propagation, ...
 - Object management

INTERNAL OF CASTOFF

Internal of CastOff[1/6]

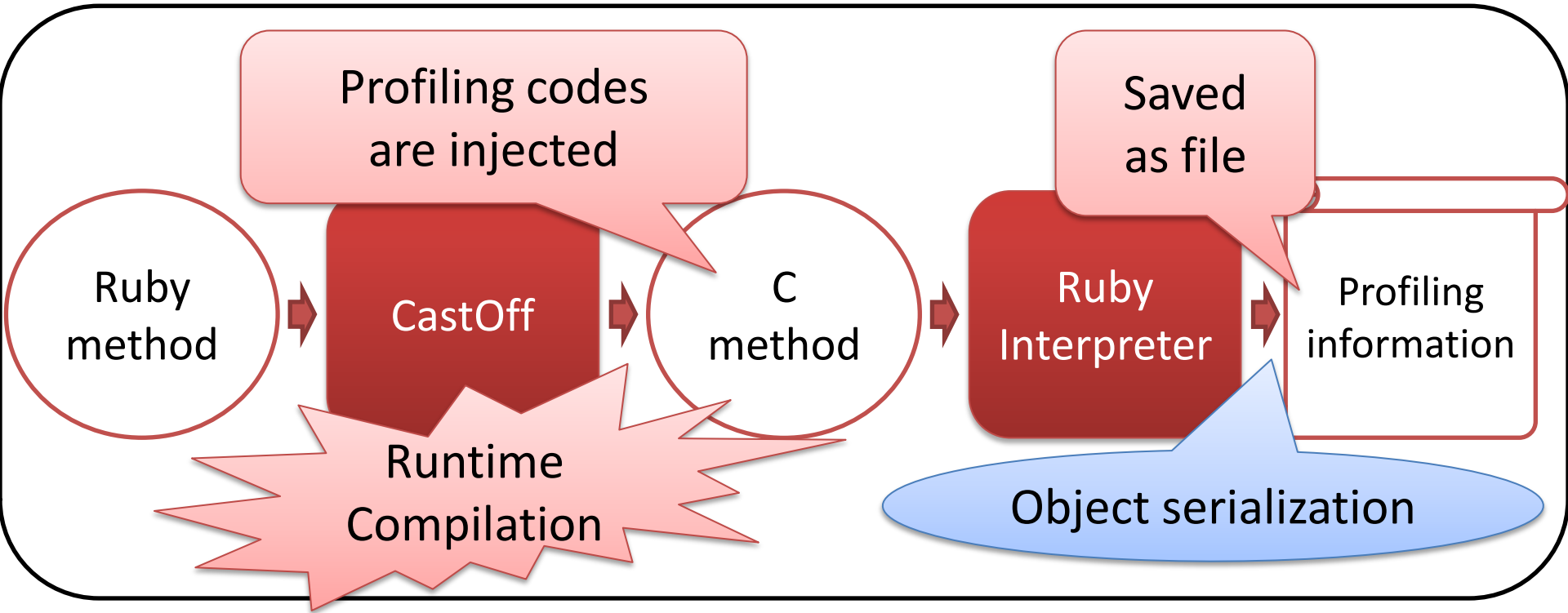
Runtime compilation



Flow of runtime compilation

Internal of CastOff[2/6]

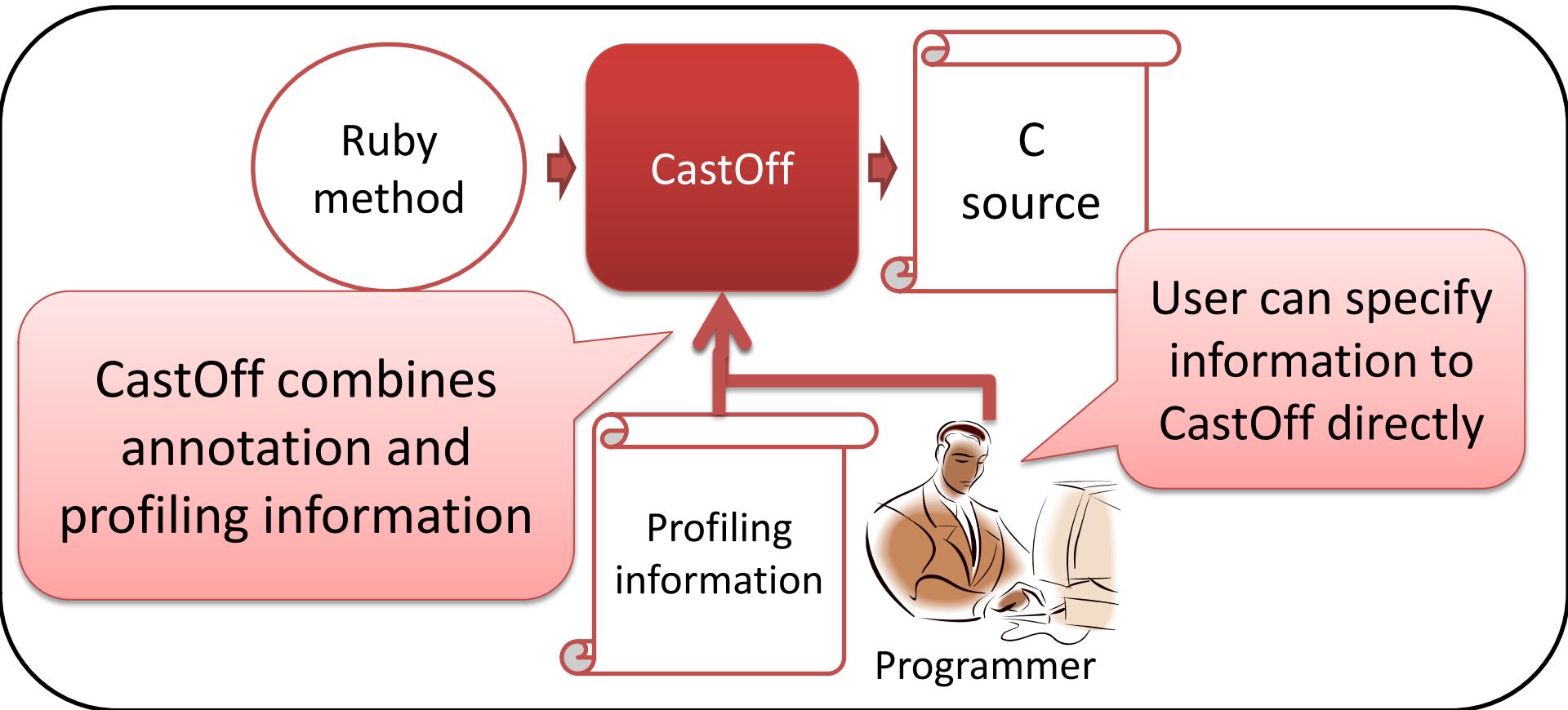
Profiling execution



Flow of profiling execution

Internal of CastOff[3/6]

Annotation support



Flow of utilizing user annotation

Internal of CastOff[4/6]

Deoptimization[1/2]

```
...  
if (!guard(local0_i, Fixnum)) {  
  recompile(sign, local0_i, "i");  
  pc = 2;  
  goto deoptimize;  
}  
...  
deoptimize:  
  context[0] = local0_i;  
  context[1] = local1_j;  
...  
original_code(context, pc);
```

Set pc

Set
contexts

Deoptimizer
of
CastOff

Call original code
with passed
pc and contexts

Original
code

Compiled code

Flow of deoptimization

Internal of CastOff[4/6]

Deoptimization[2/2]

```
...  
local_0 = fptr_Foo_bar(arguments);  
...
```

Compiled code

Method invocation
of Foo#bar

Call through
function pointer

```
...  
if (c_method(Foo, bar)) {  
  fptr_Foo_bar = get_fptr(Foo, bar);  
} else {  
  fptr_Foo_bar = method_dispatcher;  
}  
...
```

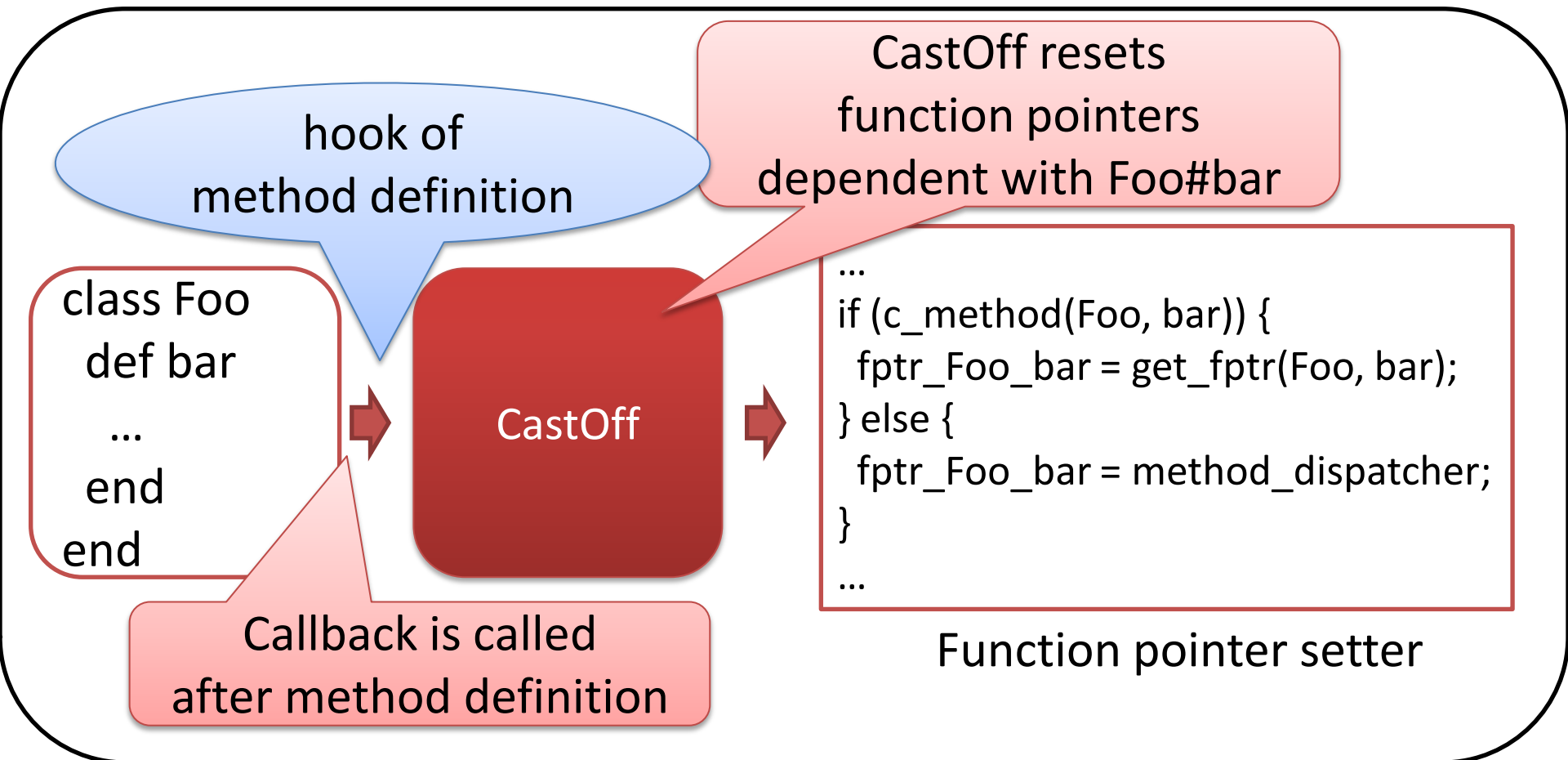
Function pointer setter

When Foo#bar is redefined,
CastOff should be
update function pointer

Flow of deoptimization

Internal of CastOff[4/6]

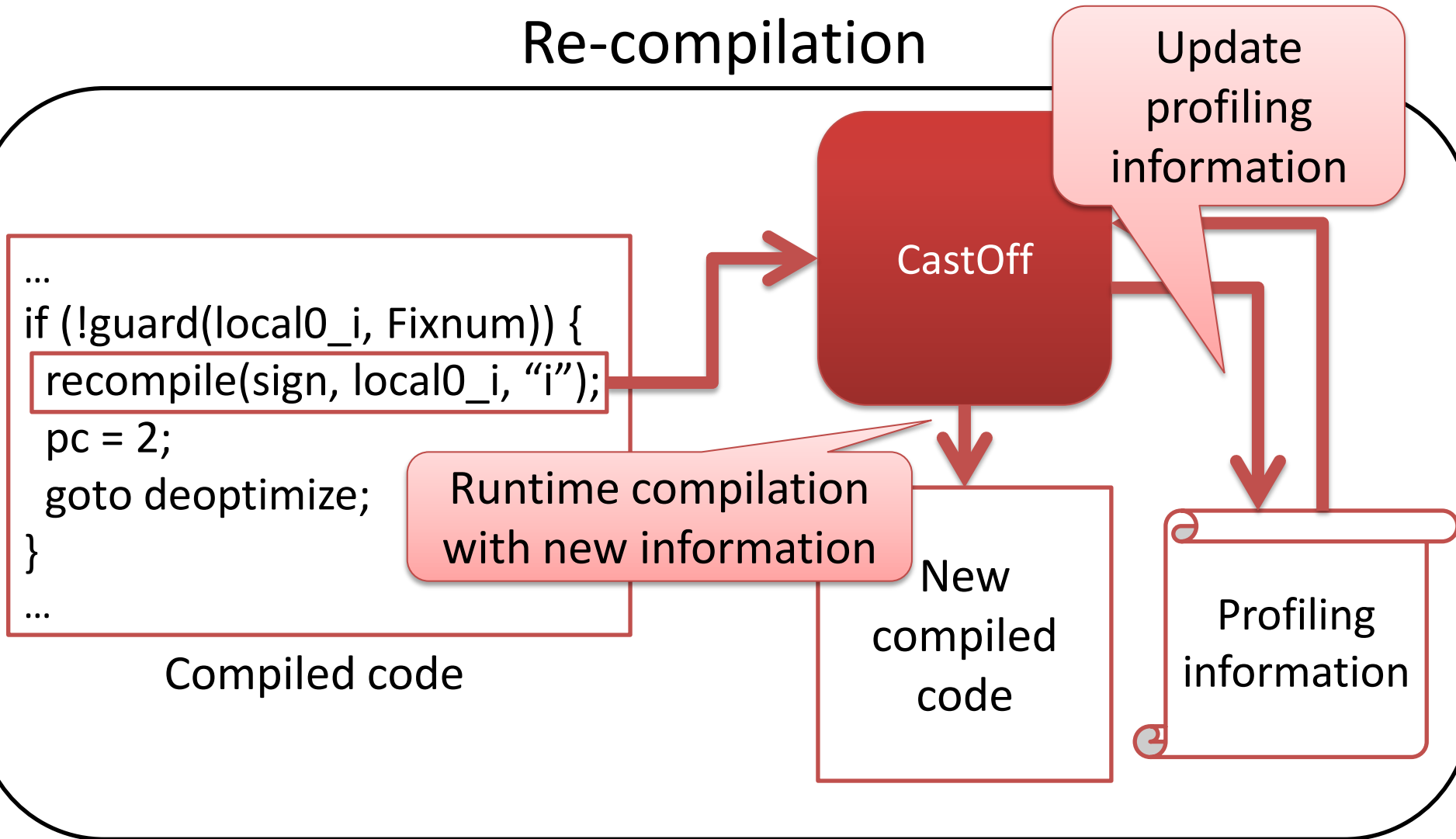
Deoptimization[2/2]



Flow of deoptimization

Internal of CastOff[5/6]

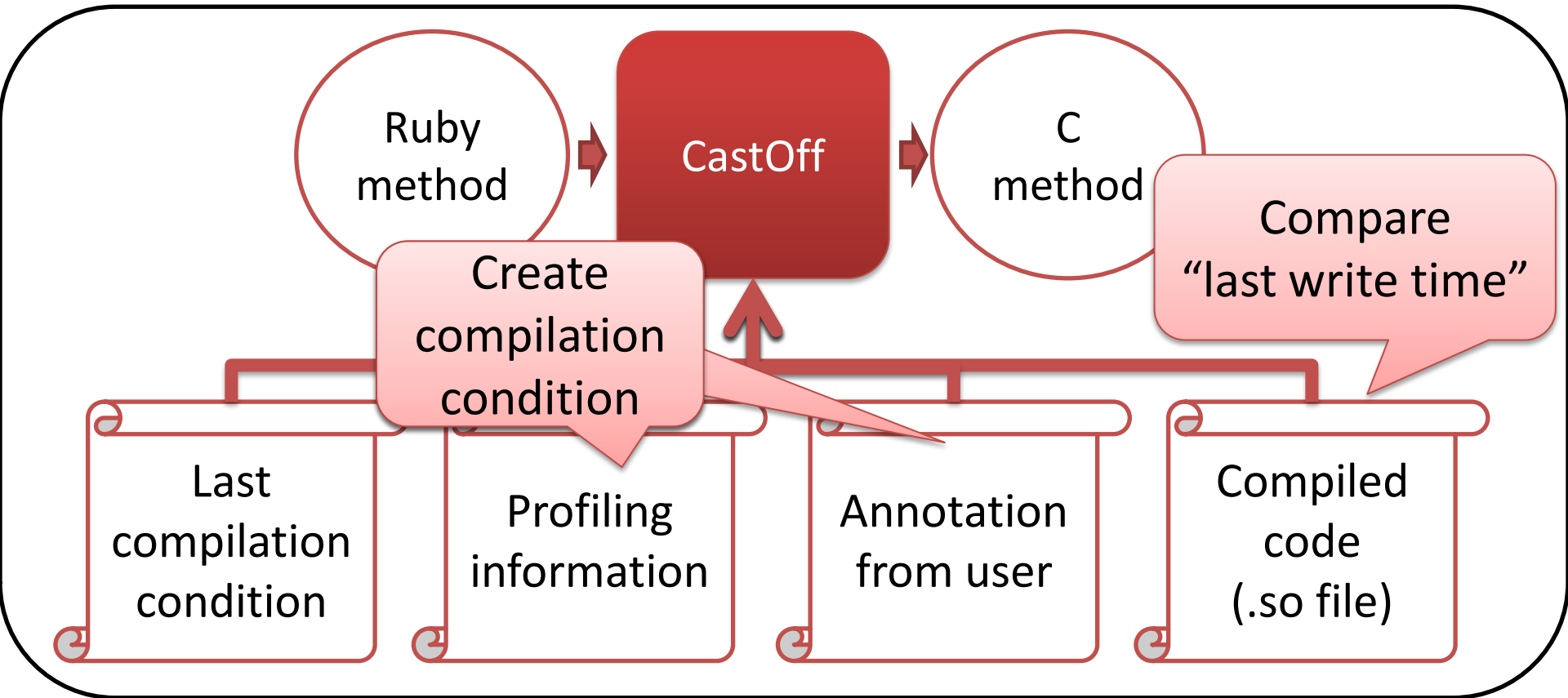
Re-compilation



Flow of re-compilation

Internal of CastOff[6/6]

Reuse of compiled codes



Flow of reusing compiled code

EVALUATION

Preliminary Evaluation

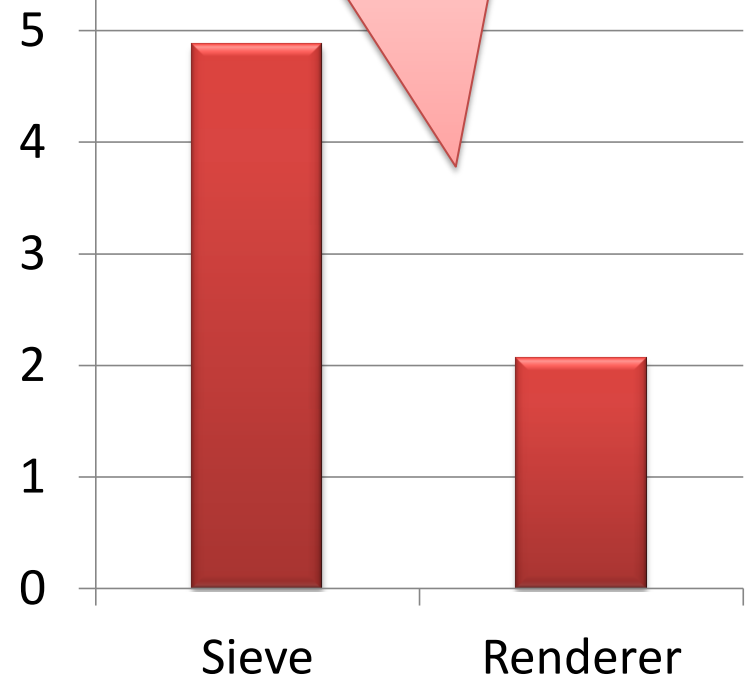
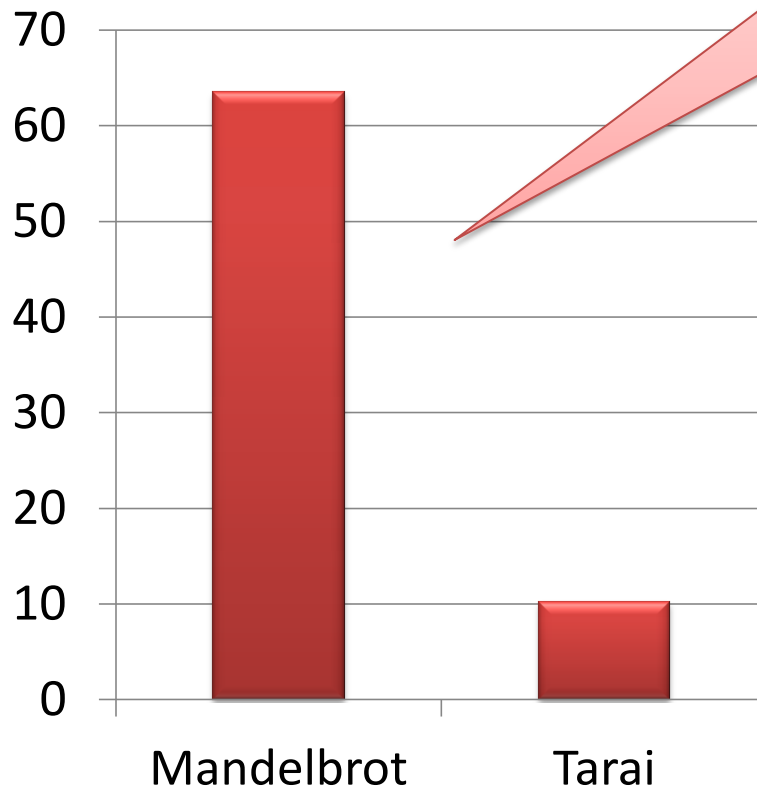
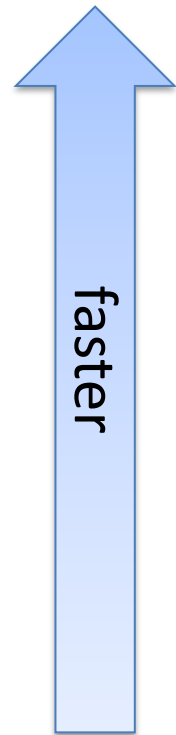
Compilation time is not included
in this evaluation

- Evaluation method
 - Compile benchmarks using CastOff **before evaluation**
 - Execute 3 times and compare minimum execution time
- Evaluation environment

Ruby interpreter	ruby1.9.3-p0
CPU	IntelCore2Quad 2.66GHz
Memory	4GB
OS	GNU/Linux 2.6.31 32-bit
Compiler	GCC4.4.1 -O3

Preliminary Evaluation

Execution time ratio (CRuby / CastOff)



Improve
performance
entirely