CastOff: A Compiler for Ruby Implemented as a Library

Satoshi Shiba
(Sasada Lab.)
2011/12/02

Abstract

- Introduce about CastOff
  - **A Compiler for Ruby Implemented as a Library**
  - Functions
    - Runtime compilation
    - Deoptimization
    - Re-compilation
    - Profiling execution
    - Reuse of compiled codes
    - Annotation support

CastOff provides these functions without modifying a Ruby interpreter

Background: Scripting Language

- Scripting languages become popular
  - Ruby, Python, PHP, JavaScript , Perl, …
  - Higher productivity
  - Lower performance

Background: Scripting Language Compiler[1/2]

- Rubinius, PyPy, hiphop-php, V8, …
  - Aiming faster execution
  - Compile into low level language(C, Assembly)
  - **Re-implementing language runtime**
    - **<u>High</u> development cost**

Background: Scripting Language Compiler[2/2]

- My master research (Compiler for Ruby)
  - Aiming faster execution
  - Compile into C
  - Almost full compatibility and High portability
  - **Utilize functions of target language runtime**
    - **<u>Low</u> development cost**
  - **Modifying target language runtime**
    - **<u>High</u> installation cost**

Reflections of my master research

Purpose and Approach

- Purpose:
  - Develop **practical** scripting language compiler easily
    - **High compatibility, High portability, Low installation cost**
- Approach:
  - For compatibility, portability and development cost
    - Utilize functions of target language

Same as master research

  - For installation cost
    - Implement compiler as library

Current challenge

Proposal

CastOff: A Compiler for Ruby Implemented as a Library

- CastOff Utilize Ruby functions
- **CastOff is Isolated from Ruby interpreter**
- **CastOff is Implemented as C extension library**

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 is hosted on Rubygems.org
  - Installation: **gem install cast_off**
  - Command line tool cast_off is available after installation

CastOff provides these functions without modifying a Ruby interpreter

## 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_singlet
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)
```

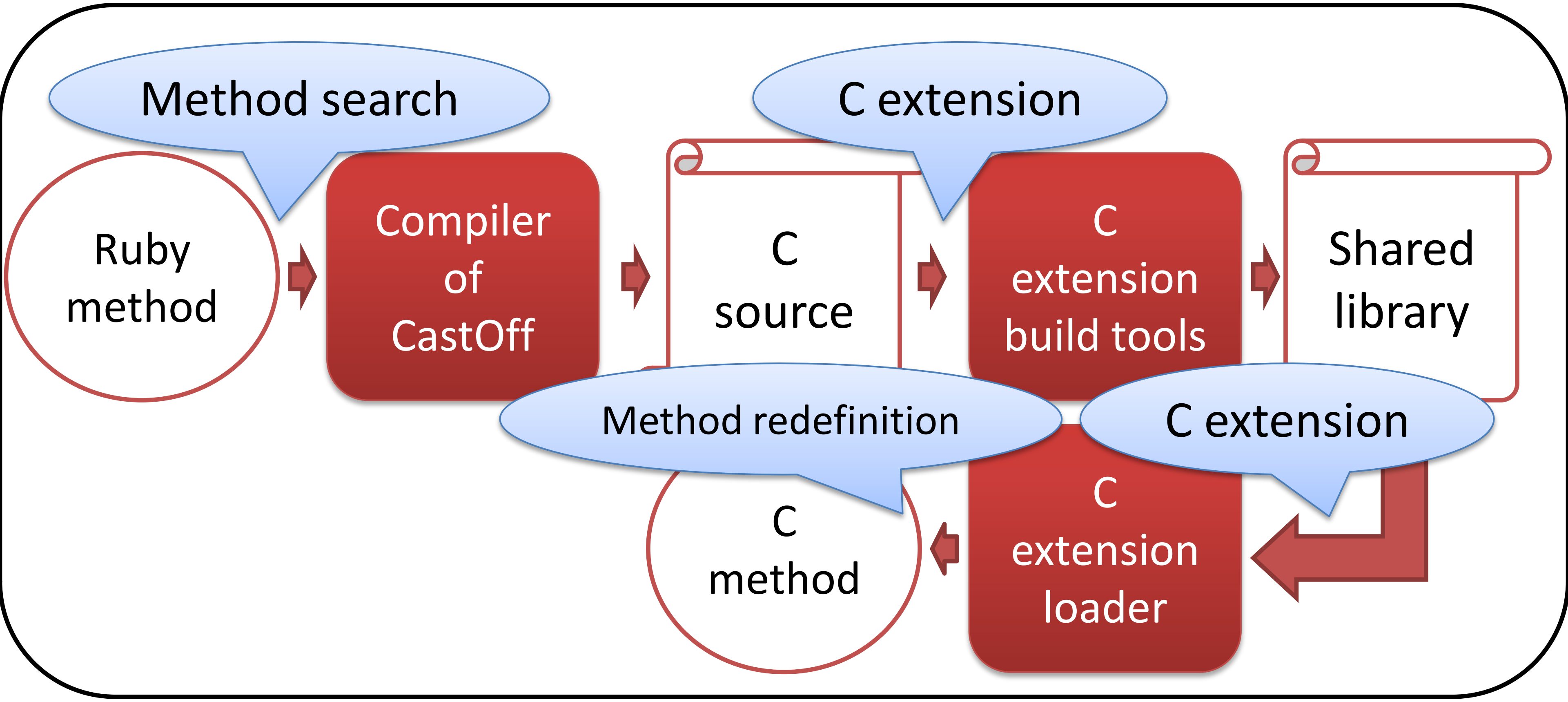
Optimization of CastOff

- Current optimization
  - Devirtualization
  - Redundant String literal duplication elimination
  - Block inlining
  - Unboxing
  - Constant prefetch
- Future optimization
  - Classical optimizations

  ⇒ Method inlining, Constant propergation, …
  - Object management

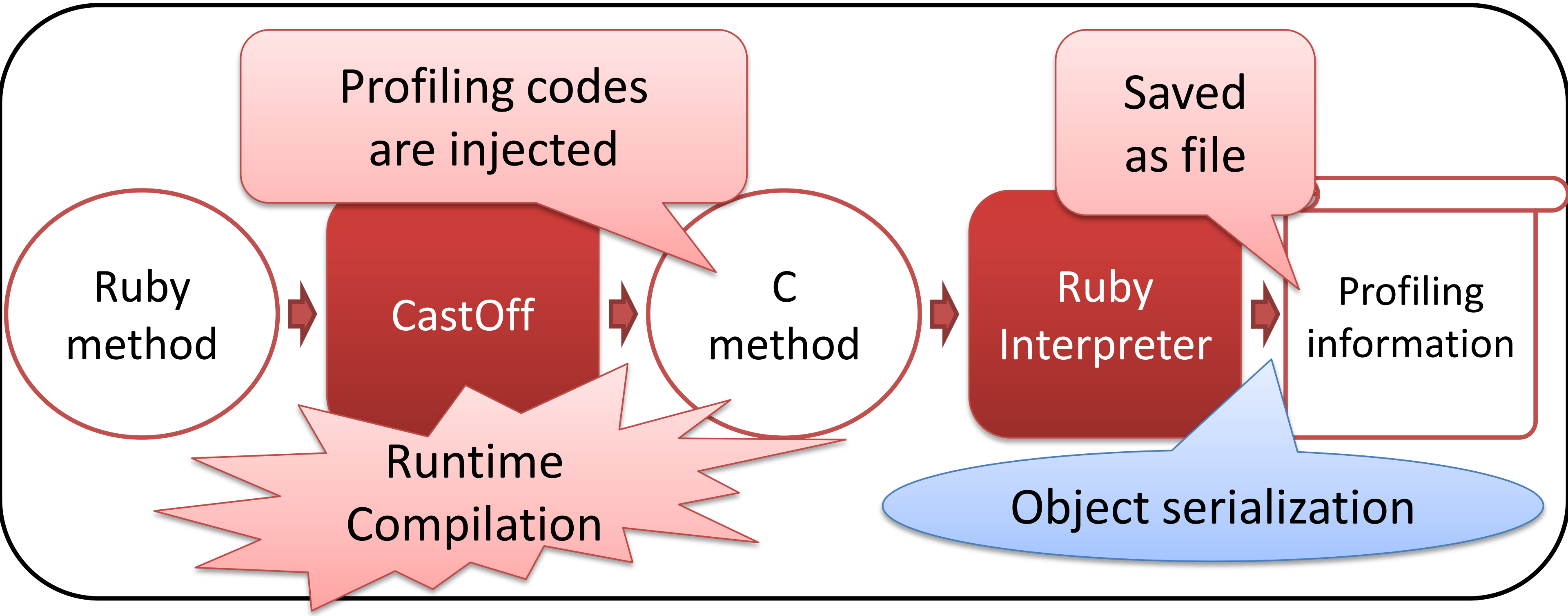
INTERNAL OF CASTOFF

Internal of CastOff[1/6]
Runtime compilation



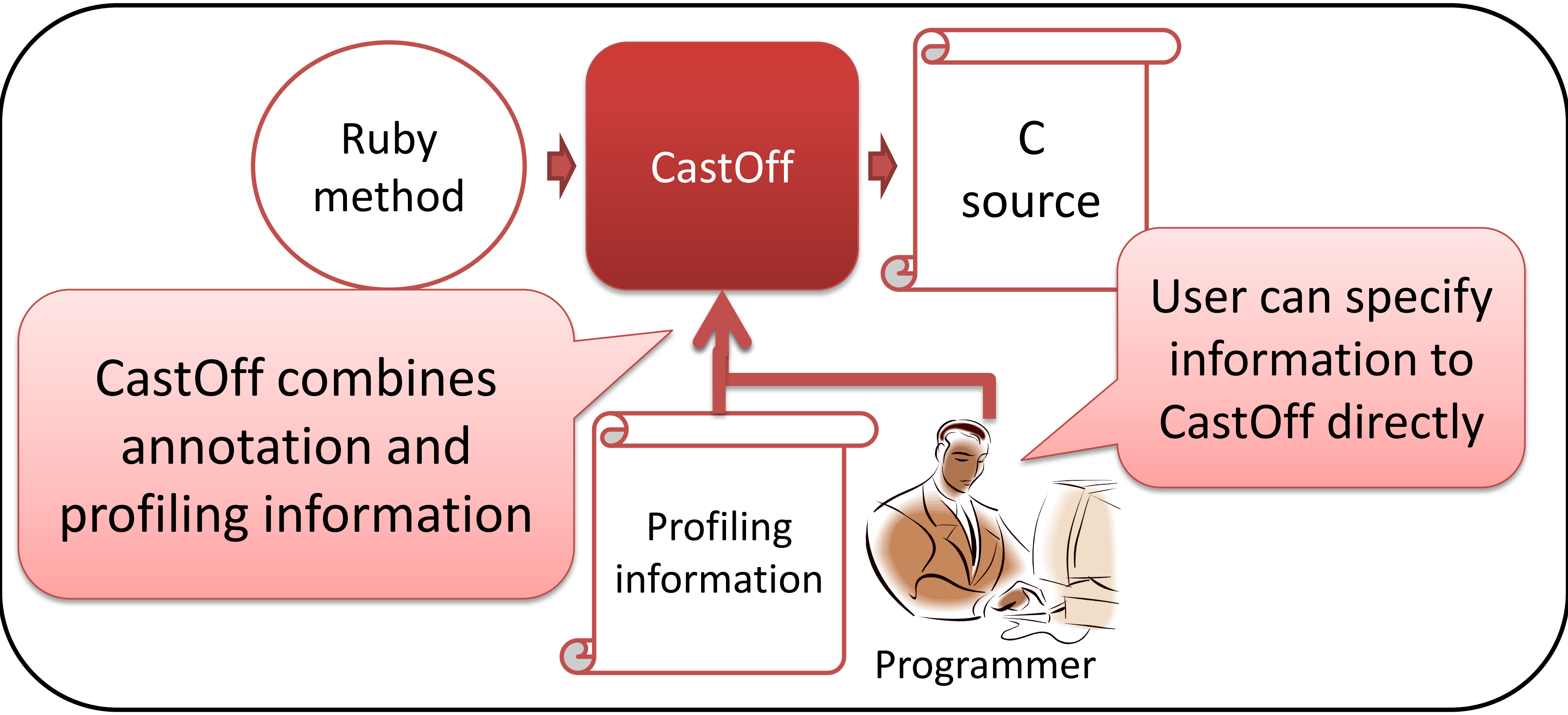
Flow of runtime compilation

Internal of CastOff[2/6]
Profiling execution



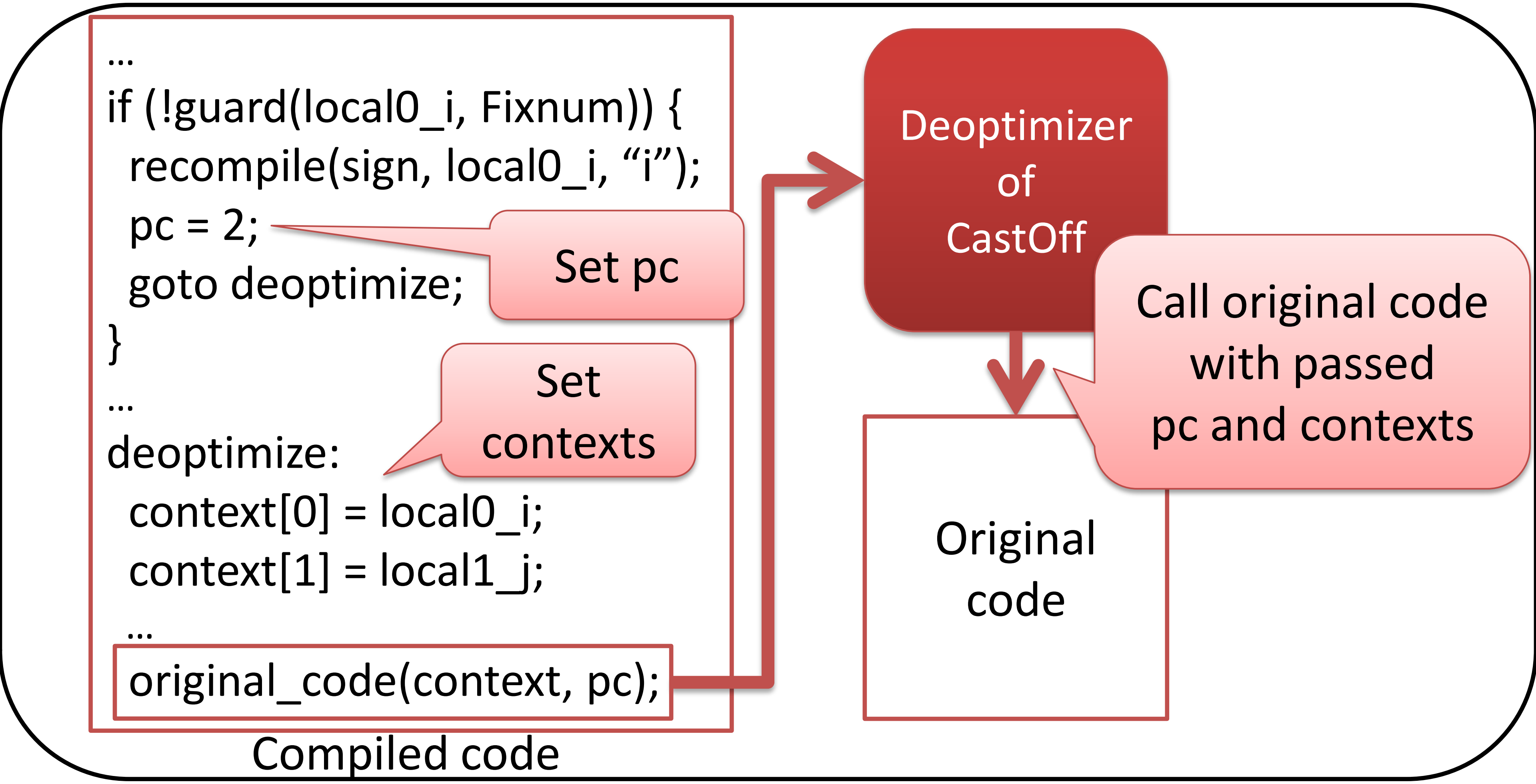
Flow of profiling execution

Internal of CastOff[3/6]
Annotation support


Ruby method
CastOff
C source
User can specify information to CastOff directly
CastOff combines annotation and profiling information
Profiling information
Programmer

Flow of utilizing user annotation

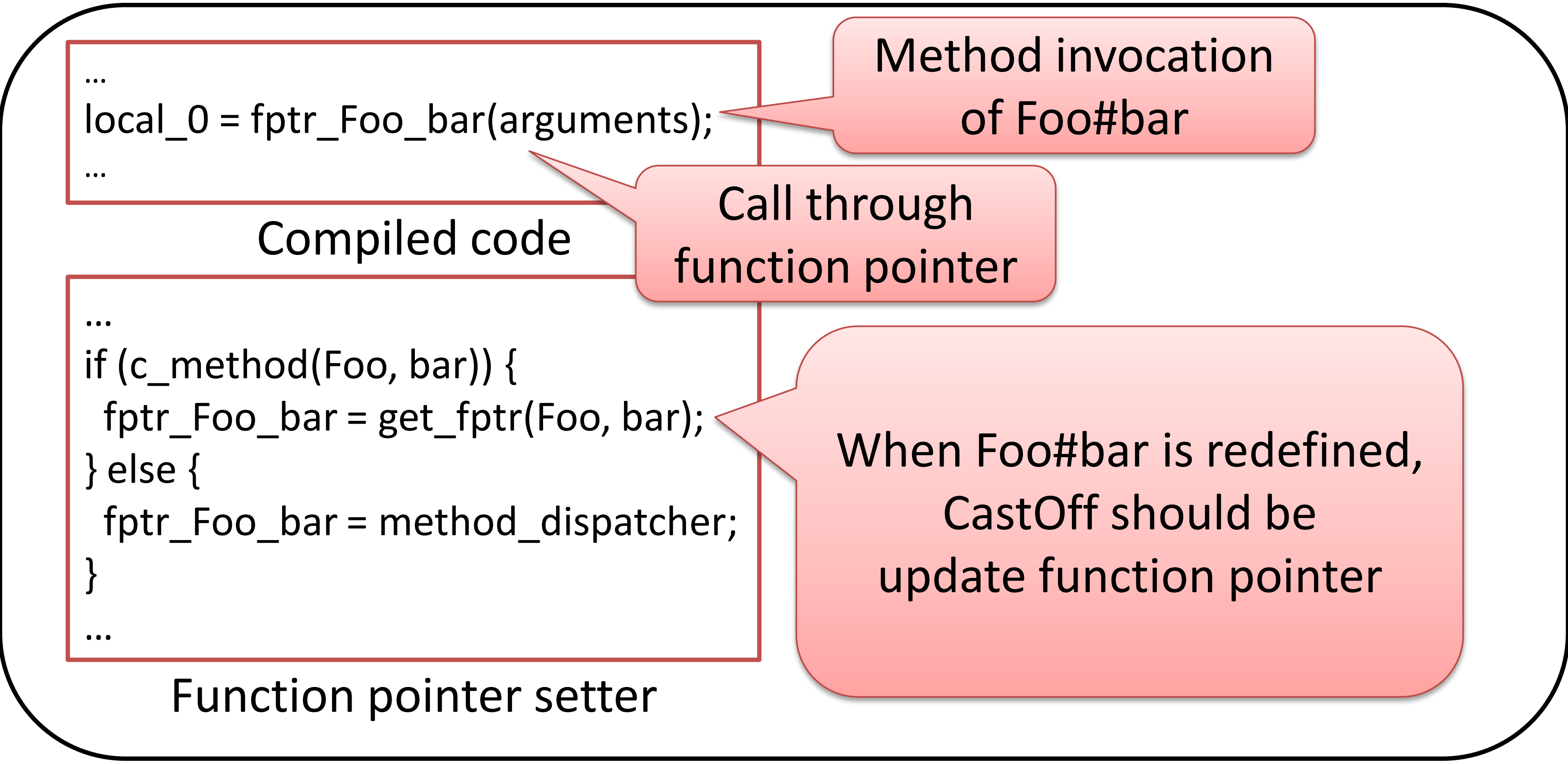
Internal of CastOff[4/6] Deoptimization[1/2]



```
…
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);
```

Flow of deoptimization

Internal of CastOff[4/6] Deoptimization[2/2]



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

Compiled code

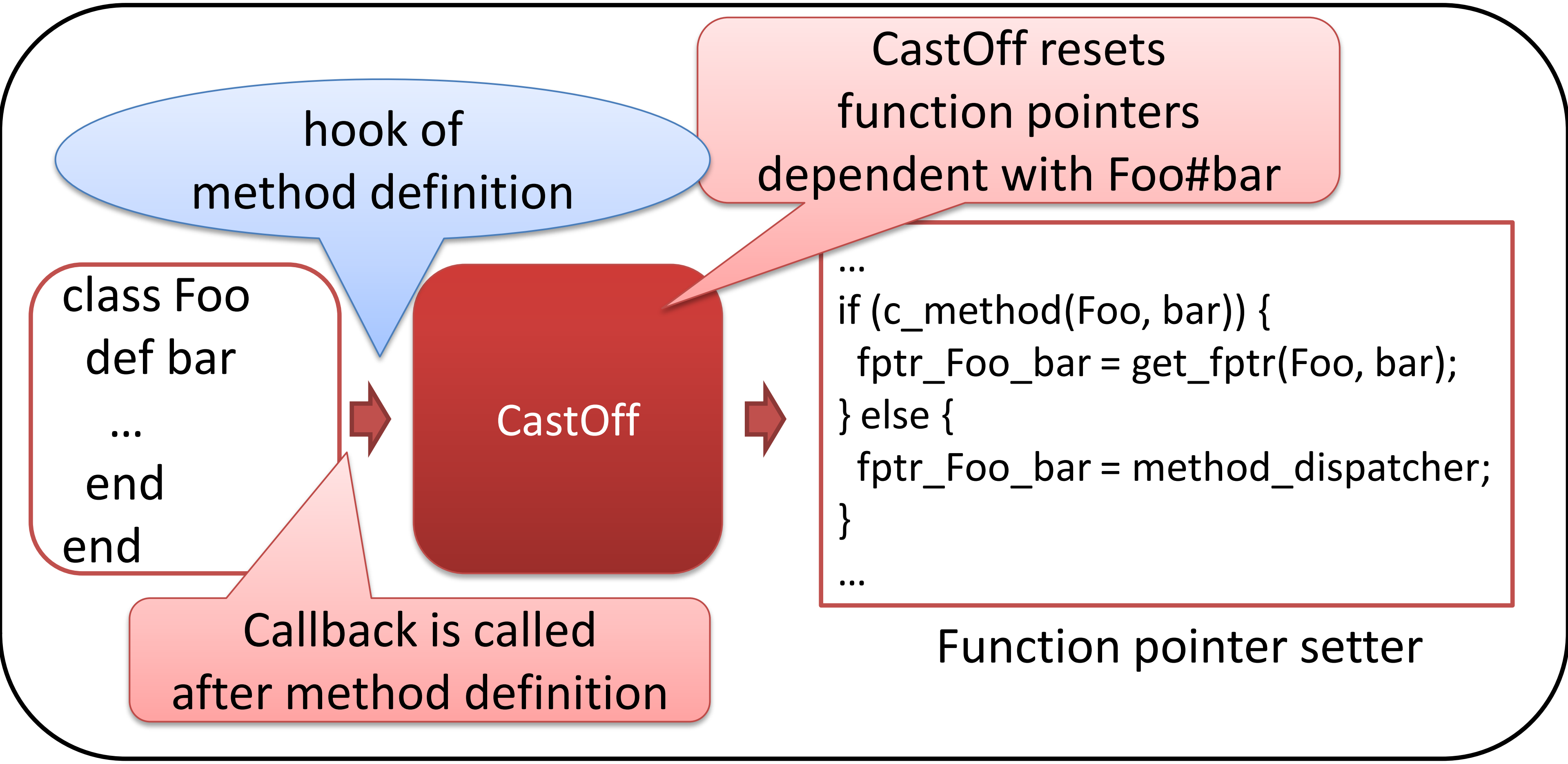
```
…
if (c_method(Foo, bar)) {
  fptr_Foo_bar = get_fptr(Foo, bar);
} else {
  fptr_Foo_bar = method_dispatcher;
}
…
```

Function pointer setter

Method invocation of Foo#bar
Call through function pointer
When Foo#bar is redefined, CastOff should be update function pointer

Flow of deoptimization

Internal of CastOff[4/6] Deoptimization[2/2]


hook of method definition
CastOff resets function pointers dependent with Foo#bar
CastOff
Callback is called after method definition
Function pointer setter

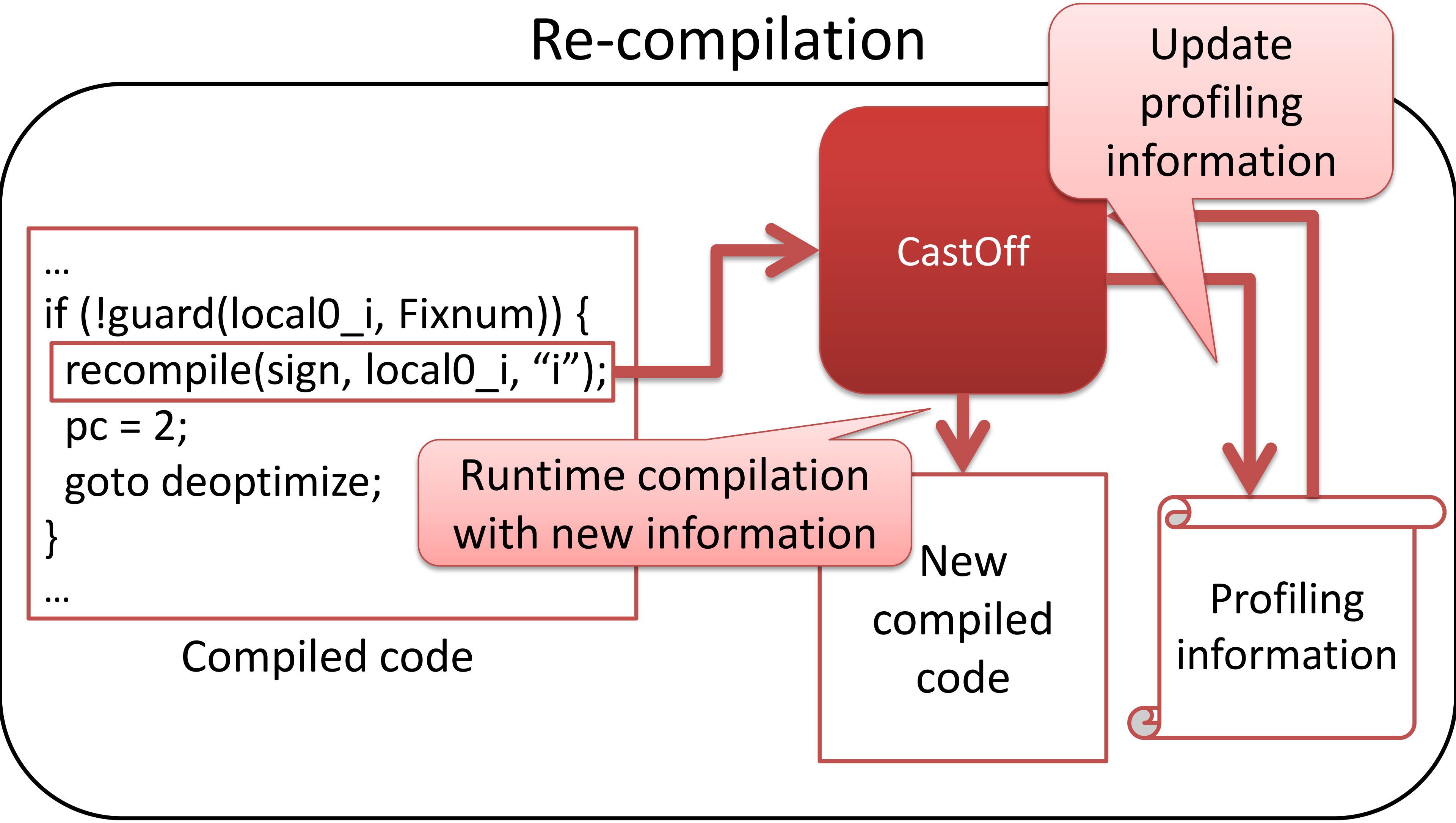
```
class Foo
  def bar
    …
  end
end
```

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

Flow of deoptimization

Internal of CastOff[5/6]

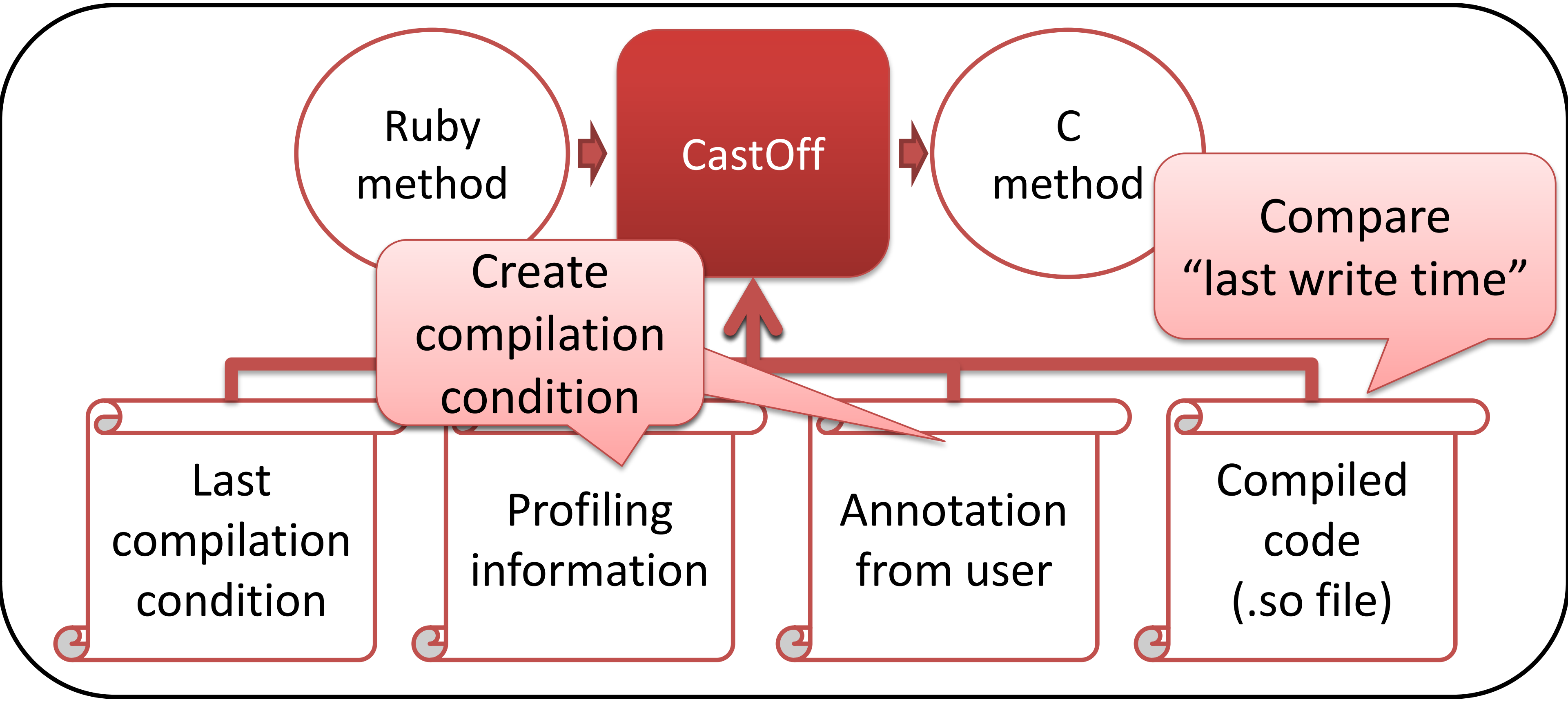
Re-compilation


Update profiling information
CastOff
Runtime compilation with new information
New compiled code
Profiling information
Compiled code

```
…
if (!guard(local0_i, Fixnum)) {
  recompile(sign, local0_i, “i”);
  pc = 2;
  goto deoptimize;
}
…
```

Flow of re-compilation

Internal of CastOff[6/6]
Reuse of compiled codes



Flow of reusing compiled code

EVALUATION

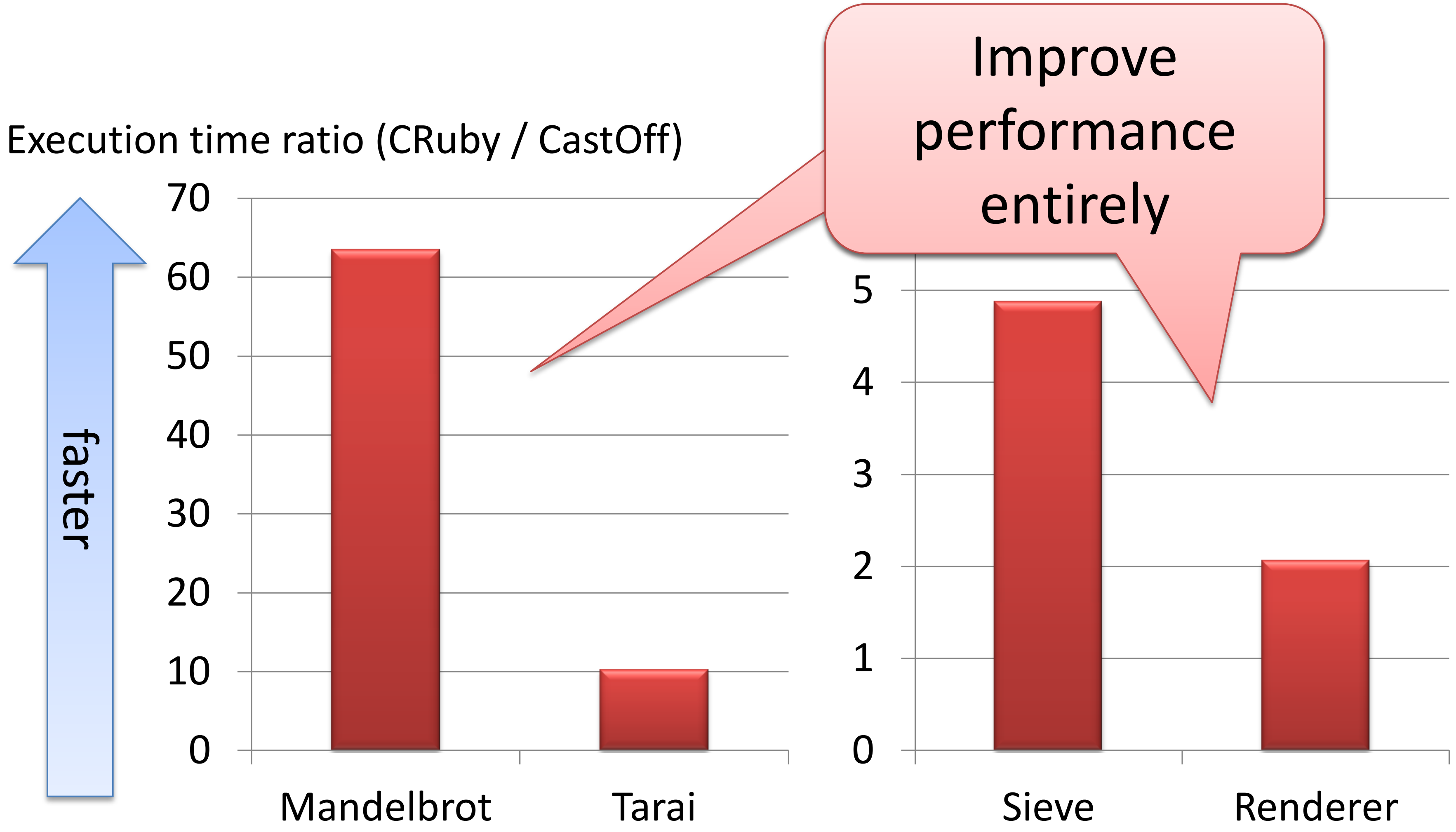
Preliminary Evaluation

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



CONCLUSION

Through development of CastOff

- **It is hard to implement compiler as library**
  - CastOff is implemented utilizing ruby functions
    - Method redefinition, Method search,
      Hook of method definition, C extension, …
  - Ruby interpreter lacks some functions for CastOff
    - **Unexported functions**
    - **Non-serializable object**
    - **No-way to hook constant re-definition**
    - **Difference between Ruby method and C method**

Current Status

- Basic functions of CastOff have largely implemented
  - Supports runtime compilation, deoptimization, …
  - Handle many Ruby functions
  - Released on Rubygems.org
- Summarizing knowledge through CastOff development
  - **Discuss how functions are needed**
    to implement compiler outer language runtime
  - Submit a paper to IPSJ PRO

Related Work

- Armin Rigo. Representation-based just-in-time specialization and the psyco prototype for python, PEPM '04 Proceedings of the 2004 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation, 2004.
  - **Compiler for Python implemented as a library**
- Biggar, Paul and de Vries, Edsko and Gregg, David: A practical solution for scripting language compilers, SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing (2009).
  - **Compiler for PHP implemented outer PHP interpreter**

•We implement compiler for Ruby
•We'll discuss functions of interpreter
to implement compiler outer interpreter

Summary

- CastOff: A Compiler for Ruby Implemented as a Library
  - Utilize CRuby functions
  - Isolated from CRuby interpreter
  - Implemented as C extension library
- Many projects re-implement language runtime
  - To improve performance
  - Large implementation cost and maintenance cost
- We implement CastOff as a C extension library of Ruby

CastOff is hosted on Rubygems.org

•Installation: **gem install cast_off**

•Command line tool cast_off is available after installation.

Thank you for your kind attention