

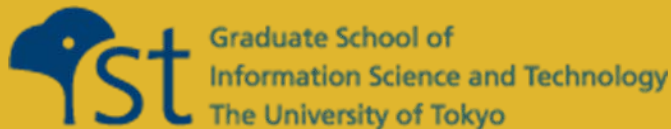


Can you see me?

Future of Ruby VM

Talk about Ruby VM Performance.

Ruby VMの未来, とかなんとか



SASADA Koichi <ko1@rvm.jp>

Department of Creative Informatics,
Graduate School of Science and Technology,
The University of Tokyo

Summary of My Talk

**“Scaling Ruby (without the Rails)”
Seems Interesting!**

**“Monkeybars: easy cross platform
GUIs” Also Does!**

On My Performance Interesting,
Former is Preferred 😊
Anyone make a Log?

Summary

CRuby/YARV is
NOT a “BEST” Solution
for Ruby VM Performance.

However, CRuby/YARV is
“GOOD” Enough Solution for Us,
the Pragmatic Ruby Programmers,
at least **Several Years.**

Self Introduction

Recent Report about Me

- ko1 - Koichi (Given Name) Sasada (Family Name)
 - From Japan, 5th RubyConf since 2004, 4th Speach
 - YARV Developer
- **Lecturer**
 - Department of Creative Informatics, Graduate School of Science and Technology, The University of Tokyo.
 - Lecture: Programming System, but only 3 students attend
- **SASADA-lab**
 - If you want to research about Ruby or Virtual Machine, Systems Software **in Japan**, please contact me.
 - 2 students are there, but no one want to hack YARV.

Caution! (re-re-review)

- I can't speak English well
 - If I say strange English, you can see the slide page
 - Or ask another Japanese. They can speak English well.
 - My Slides uses Small Characters (against Takahashi-san's Presentation Method)
 - If you have any question, ask me with:
 - Japanese (recommended)
 - **Ruby**, C, Scheme, Java, ..., Python, Haskell, ...
 - Or Easy English

Agenda

- Perspective of Ruby VM Performance
 - VM Performance Discussion
 - Our Performance Policy
- Introduction of Our Research
 - Hidden Optimization Techs.
 - Ricsin Project
 - Ruby to C AOT Compiler Project
 - atomic-Ruby Project
 - MVM Project
- Summary

Remember The Evan's Classification

JRuby is for Java Programmers

IronRuby is for .Net Programmers

Rubinius is for Ruby Programmers

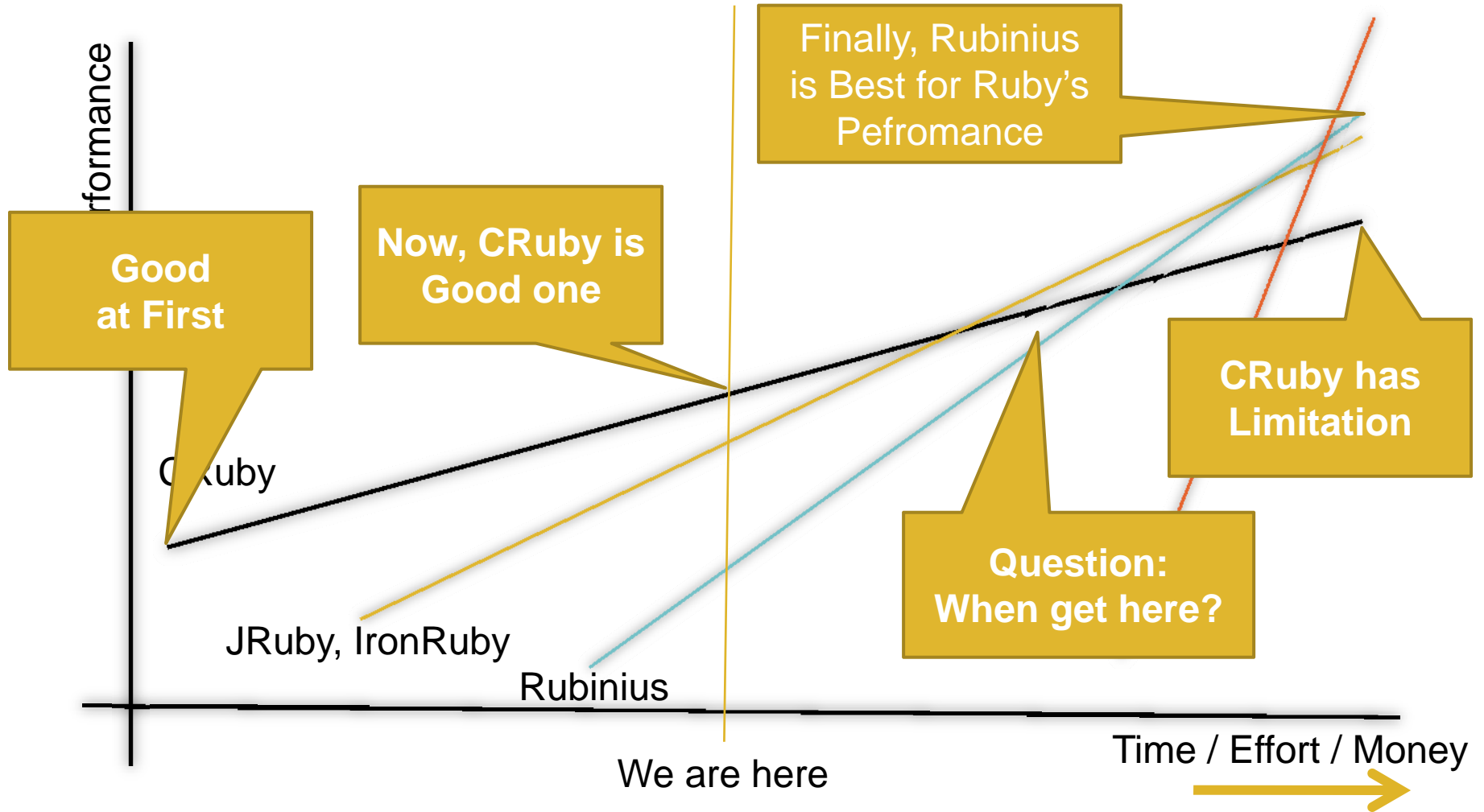
CRuby is for C Programmers



OK.

Let's Talk about the "C",
The Benefits and Limitation

Evolution of VM Performance My Prediction



Techniques for VM Performance

- Simple Optimization Techniques
 - C-level VM Techniques
- Advanced Optimization Techniques
 - Dynamic Code Generation
 - Speed-up using Native Machine code Compiler
 - Just in Time Compilation
 - Polymorphic Inline Cache
 - Selective Inlining
 - Online Feedback Optimization
 - HotSpot JIT Compiler
 - Tracing JIT

Pros and Cons of JRuby/IronRuby

- Using Awesome VM
- Pros.
 - Many Clever People Working on each VM
 - No Code is Good Code.
 - No Bugs are Generated.
 - Many Libraries on Each Environments
 - Easy (?) to Use Parallelization
- Cons.
 - Not Only Focused on Ruby, Semantics Gap
 - Can't Use C Extensions Directly

Pros and Cons of Rubinius

- Most of Code is Written in Ruby
 - Like Java
- Pros.
 - Ruby in Ruby
 - Meta-Circular Interpreter
 - **Best Way** to Improve Performance in the Long Run Because They Can Analyze Most of Programs.
 - Mainly Focus on Ruby
- Cons.
 - **Long Way** to Get High Performance VM

Pros of "C" Ruby

- **Portability**
 - Most of Environments have GCC Porting.
- **Maintainability**
 - Everyone Know C.
- **Extensibility**
 - Easy to Write Extension with C.
- **Performance Improvement**
 - Easy to Write Simple (Machine Independent) Optimization.

Cons (Limitation) of “C” Ruby

- C Extension Libraries or Methods written in C
 - GC Problem
 - Conservative Mark & Sweep Stop The World GC
 - Inlining Problem
 - Can't Inline C code into Ruby Code
 - Limitation of Program Analysis

Our Performance Policy

- CRuby is Not “Best” Solution but “Good” One
- Continue to Improve CRuby’s Implementation
 - in C
 - in Machine Dependent Way
- Pragmatic, Practical Selection
 - at least several years

Keywords for Success

- “Embedding”
- Parallelization

Introduction of Our Research

- To Take Advantage of "C", Some Projects are Running
 - Hidden Optimization Techs on YARV
 - Ricsin: Mix-in C to Ruby Project
 - Ruby to C AOT Compiler Project
 - atomic-Ruby Project
 - Multi-VM Project

Hidden/Left Optimization Techs

- Turned Off on 1.9.1 by Default
 - Tail call Optimization
 - Optimization using Unification
 - Stack Caching
- Left Easy Optimization
 - Efficient Method Caching
 - Efficient Fiber Implementation using Platform dependent way such as `makecontext()`
- These Optimizations will be Merged into 1.9.2

Ricsin: Mix-in C to Ruby

- Embed a part of C Program into Ruby
- Like an RubyInline, but Embed Directly
- Usage Example
 - Use C Libs Directly
 - Replace All Built-in Classes/Methods
 - Test Ruby C APIs
 - Performance Improvement Continuously

Ricsin Notation

```
def open_fd(path) # Ruby
```

```
  fd = __C__(%q{
```

```
    /* C */
```

```
    return INT2FIX(open(RSTRING_PTR(path), O_RDONLY));
```

```
  })
```

```
  raise 'open error' if fd == -1
```

```
  yield fd
```

```
ensure
```

```
  raise 'close error' if -1 == __C__(%q{
```

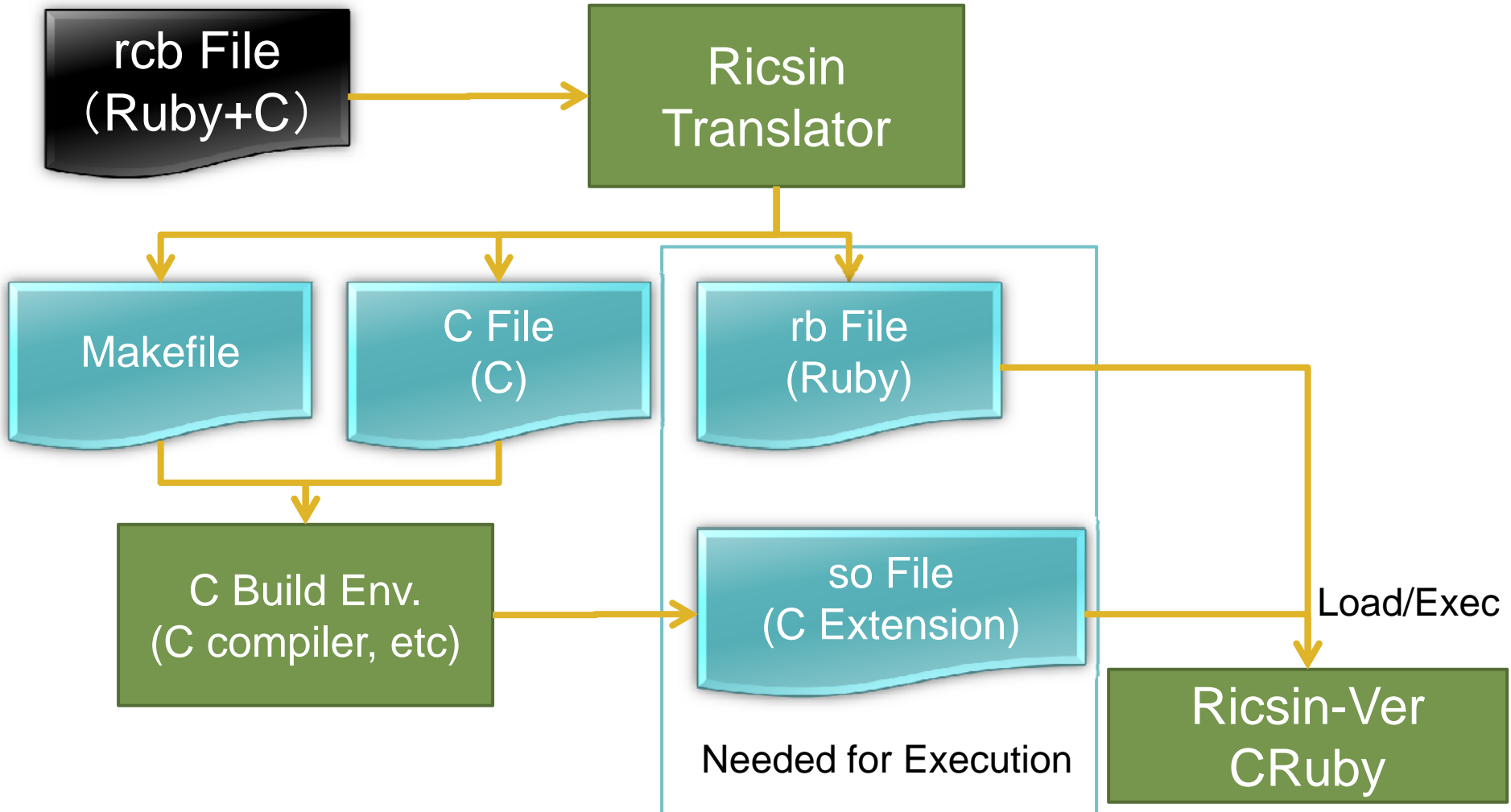
```
    /* C */
```

```
    return INT2FIX(close(FIX2INT(fd)));
```

```
  })
```

```
end
```

Ricsin Total View



Ricsin

Translation and Execution

```
# rcb
v = 42
r = __C__(%q{
  /* Embed C Body */
  rb_p(self); /* show "main" */
  return INT2FIX(
    FIX2INT(v) + 1);
})
p r #=> show "43"
```

Generate



```
/* A Part of Generated C Source */
#define v (cfp->lfp[3])
#define r (cfp->lfp[2])
VALUE ricsin_func_1(
  rb_control_frame_t *cfp)
{
  const VALUE self = cfp->self;
  {
    /* Embed C Body */
    rb_p(self);
    return INT2FIX(FIX2INT(v) + 1);
  }
  return Qnil;
}
#undef v
#undef r
```

Bytecode Compile

Function Call

[ADDR]	[INSN]	[OPERAND]
0000	putobject	42
0002	setlocal	v
0004	opt_call_ricsin	<funcptr>
0006	setlocal	r
0008	putnil	
0009	getlocal	r
0011	send	:p, 1
0017	leave	

Built to Extension Library

Ricsin: Evaluation

- Performance Evaluation (Not a Usability)
- Evaluation Environment
 - Env.1 : Intel Xeon E5335, Linux
 - Env.2 : SPARC T2, SunOS 5.10
- Evaluation Items
 1. Calling C Function (null call)
 2. Example on Iterator
 3. Matrix Multiprior

Ricsin

Evaluation of Calling Null Function

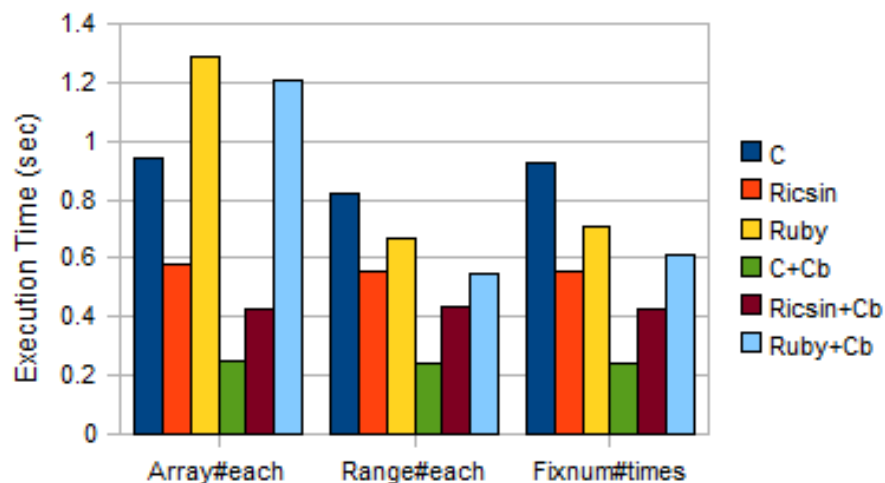
- Calling Null C Function
 - Null C Method
 - Null `__C__` Embed

	C (sec)	Ricsin (sec)	C/Ricsin
Env.1 (Intel)	0.44	0.05	8.8
Env.2 (SPARC)	4.56	0.44	10.4

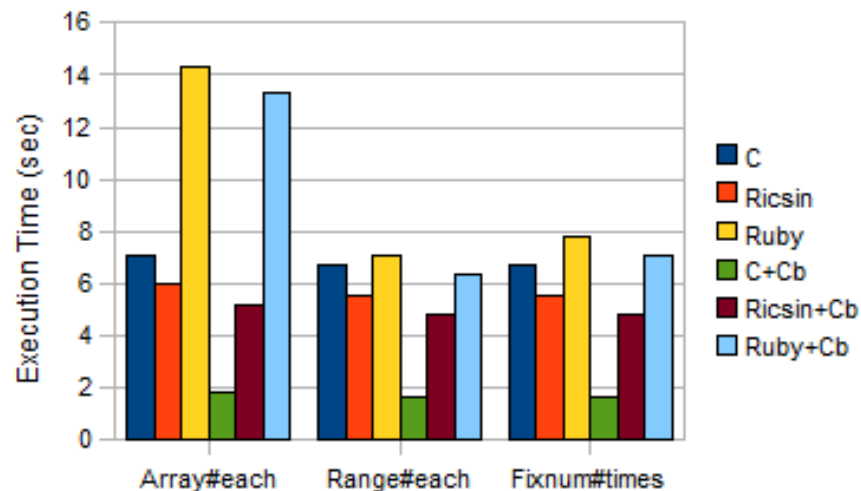
Ricsin

Evaluation: Iterator Optimization

- Rewrite Iterators with Ricsin
 - C: Current Iterator
 - Ricsin: Rewriting with `__Ccont__`
 - Ruby: Rewriting with Pure Ruby



Env.1 (Intel)



Env.2 (SPARC)

Ricsin

Evaluation: Matrix Multiplier

- Matrix Multiplier with Fixnum Elements
- Replace 12 Lines Ruby Code to 36 Lines C Code Directly

	Ruby (sec)	Ricsin (sec)	Ruby/Ricsin
Env.1 (Intel)	10.57	0.57	20.33
Env.2 (SPARC)	85.31	6.73	12.68

Ricsin

svn co

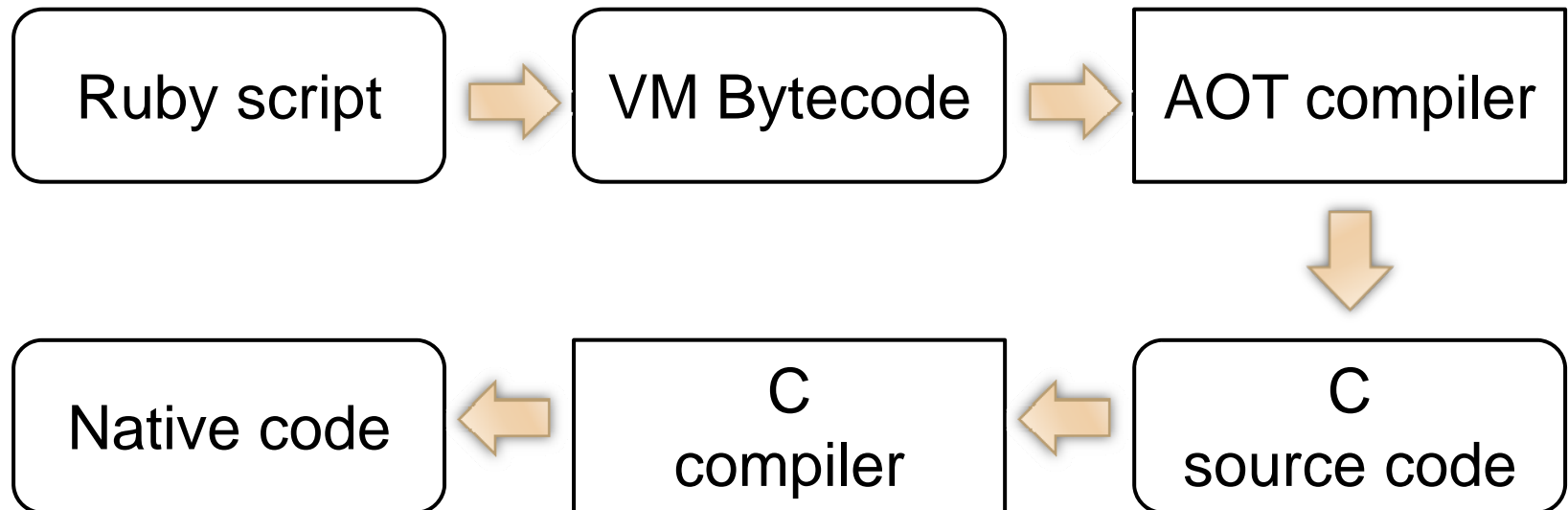
[http://svn.ruby-lang.org/
repos/ruby/branches/ricsin](http://svn.ruby-lang.org/repos/ruby/branches/ricsin)

Ruby to C AOT Compiler

- Translate Ruby Script to C Source Code at Ahead of Time
 - Compile Ruby to Bytecode
 - Translate Bytecode to C Source Code
- Performance Improvement by
 - Eliminate VM Instruction Dispatch
 - Optimization by C Compiler
 - Eliminate Parse/Compile Time

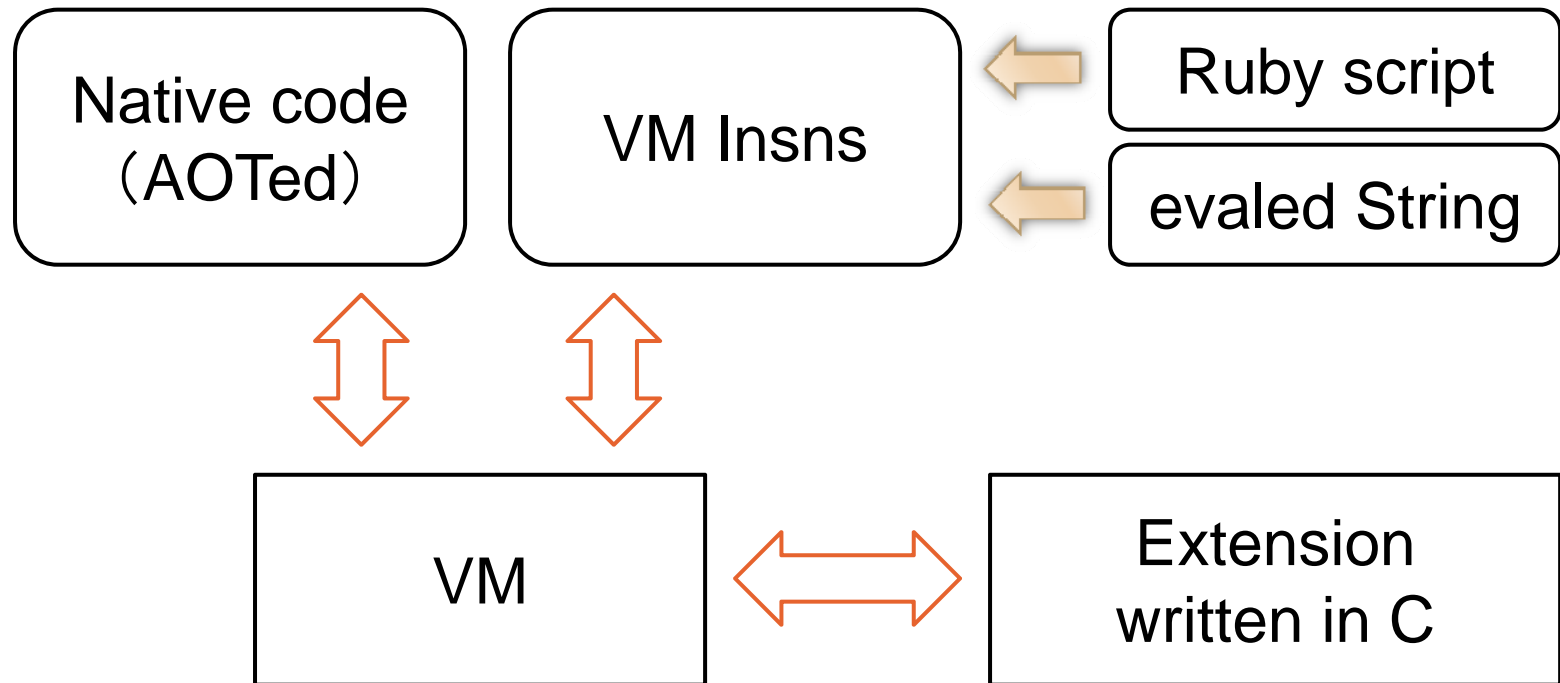
Ruby to C AOT Compiler

- Ahead of Time Compilation
 1. Compile Ruby Script to VM Bytecode
 2. VM Bytecode to C



Ruby to C AOT Compiler

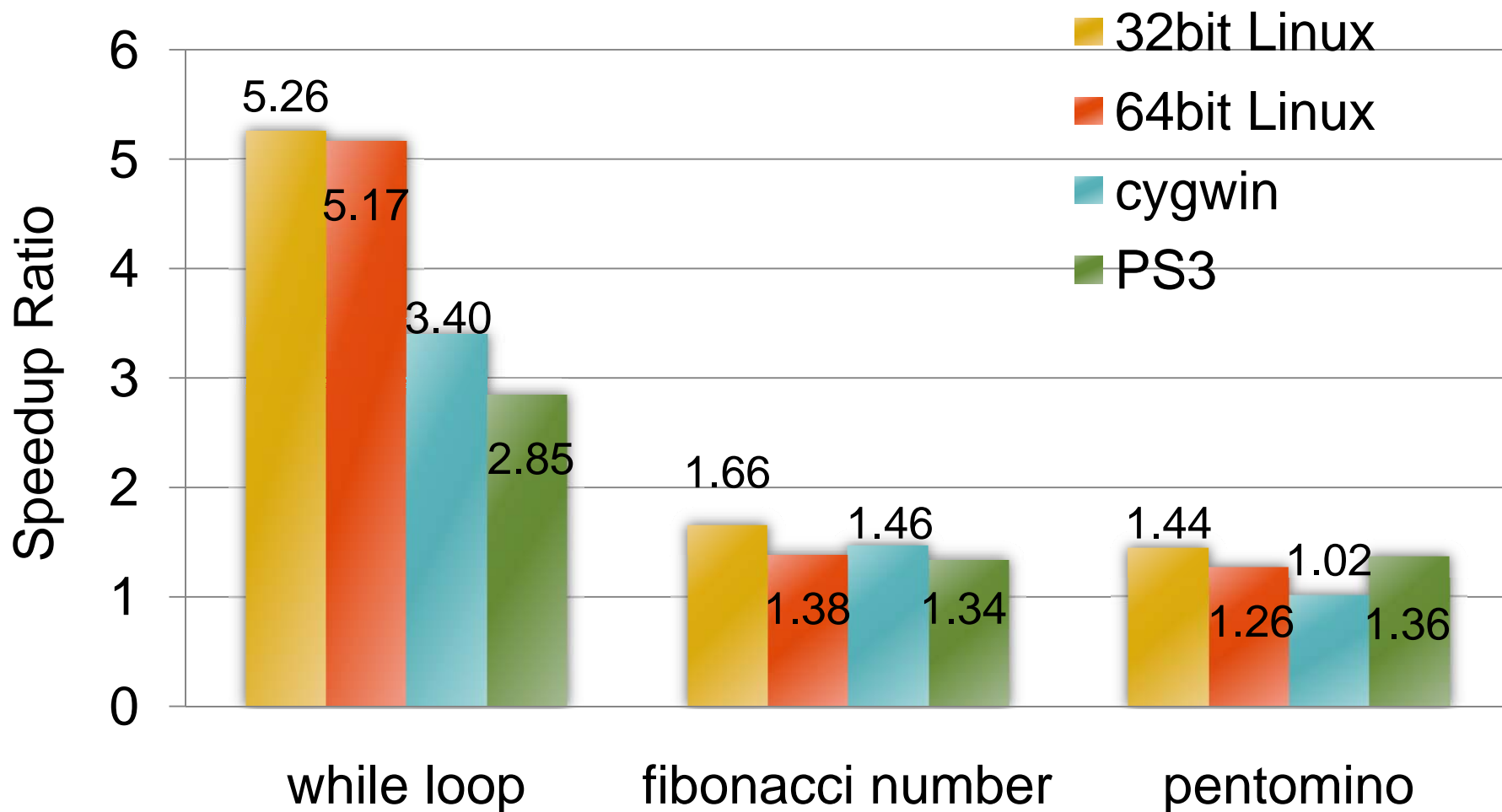
- Execution with Ruby VM



Evaluation Environment

Env	CPU	Memory	OS	C Compiler
32bit Linux	Intel PentiumD 2.80GHz	2 GB	Linux 2.6.24	gcc 4.2.3
64bit Linux	Intel Xeon 3060 2.40GHz	1 GB	Linux 2.6.18	gcc 4.1.2
cygwin	Intel Core Duo U2400 1.06GHz	1.5 GB	Windows Vista SP1	gcc 3.4.4
PS3	Cell Broadband Engine 3.2GHz	256 MB	Linux 2.6.16	gcc 4.1.1

Ruby to C AOT Compiler Evaluation Results



Related Work

- ruby2c by Eric, Ryan
 - Subset Ruby to C
- yajit by Shinh
 - JIT (yarv bytecode to IA-32 with Xbyak)
- yarv2llvm by Miura-san
 - JIT (yarv bytecode to LLVM asm)

atomic-Ruby Project

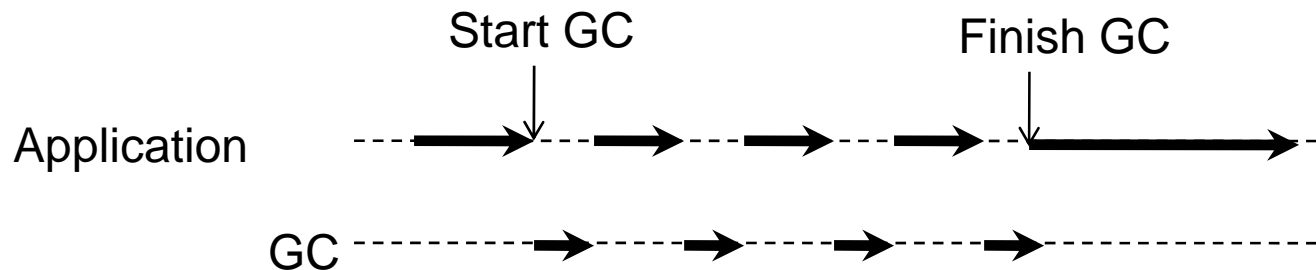
- Issue: Ruby is too Fat
 - Involves Convenient Functions.
 - Complex and Rational will be Built-in at Ruby 1.9
→ Difficult to Use “Embedded” Environment
- **“Embedded”**
 - Embedded System such as Resource Limitation Devs.
 - In Many Case, Numeric Tower or m17n are not needed.
 - Application Embedded Ruby
 - Application needs “DSL Engine”, doesn’t Full-set Ruby

atomic-Ruby Project (cont.)

- We Need Slim Ruby Interpreter
- atomic-Ruby makes “Suitable Ruby Interpreter”
 - Ruby Interpreter for Application
 - Ruby Interpreter for Environment (such as Embedded Systems)
 - Ruby Interpreter for Driver Application
- Utilize CRuby’s Portability
- 3 Sub-Project with 3 Students
 - Plug-in/out Built-in Classes/Methods
 - Pre-Compilation and Remove Parser/Compiler
 - Switch Core-Feature such as GC, Regex, Thread, etc

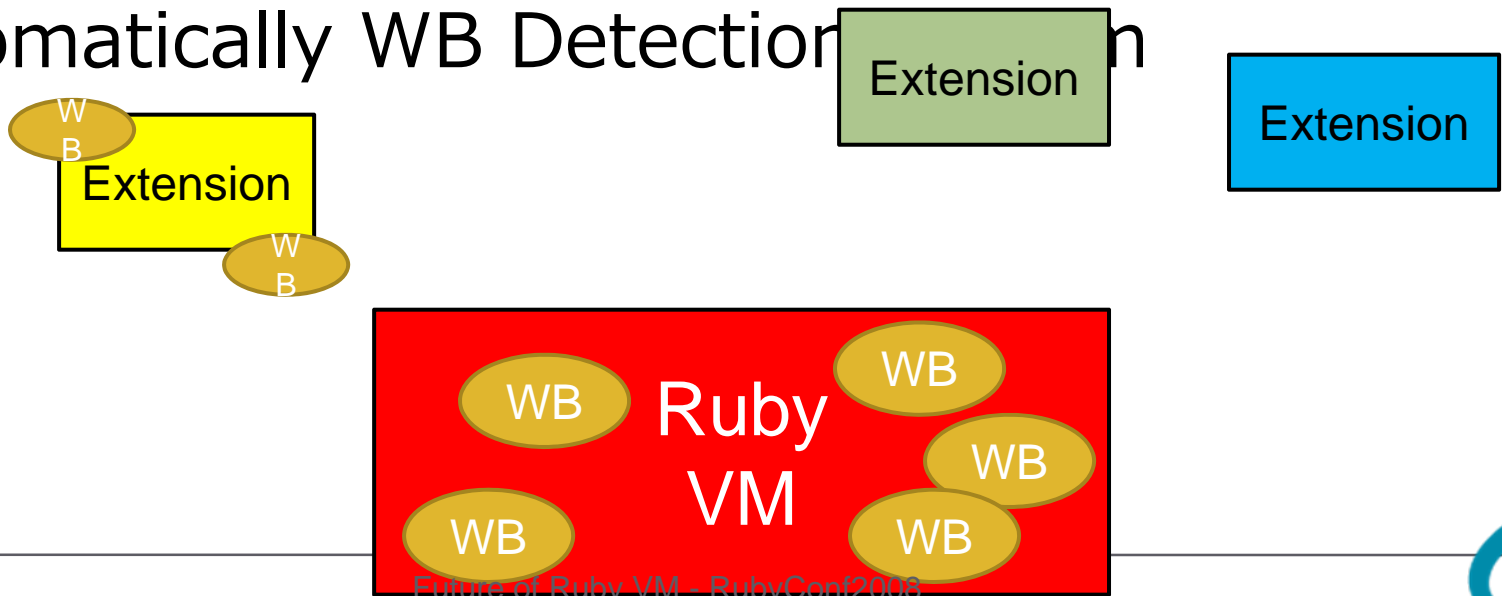
atomic-Ruby Incremental GC

- Switch GC Algorithm
- Mark Partially
 - Execute App and Mark partially
 - Reduce Application Stop Time



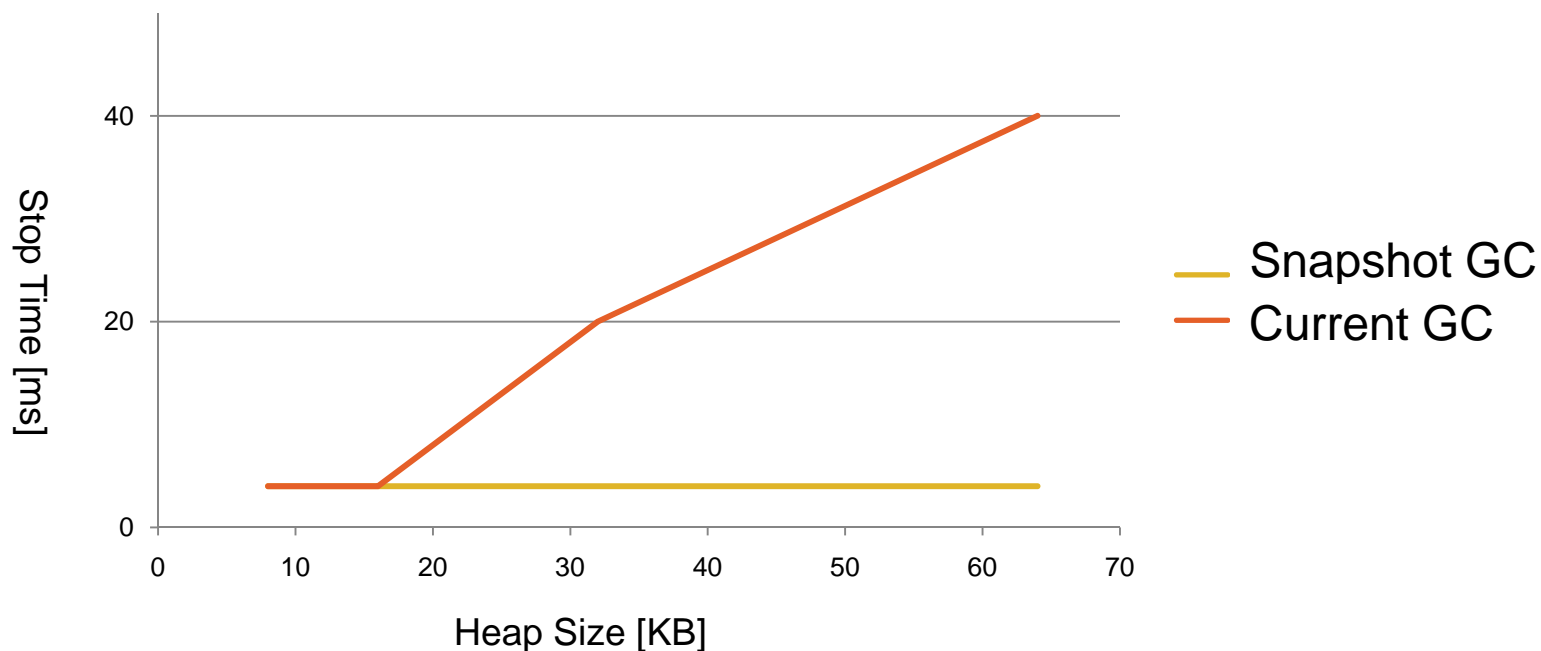
Auto Write Barrier Detection

- Write Barrier is Needed for Several GC Algorithms.
 - Need Interpreter and Extensions.
 - Need Special Knowledge of VM and GC.
 - Cause Critical Bugs if WB Insertion Miss.
- Automatically WB Detection in



Snapshot (Real Time) GC

- Stop Time of Application (Mark Phase)
 - Insert Many WBs.



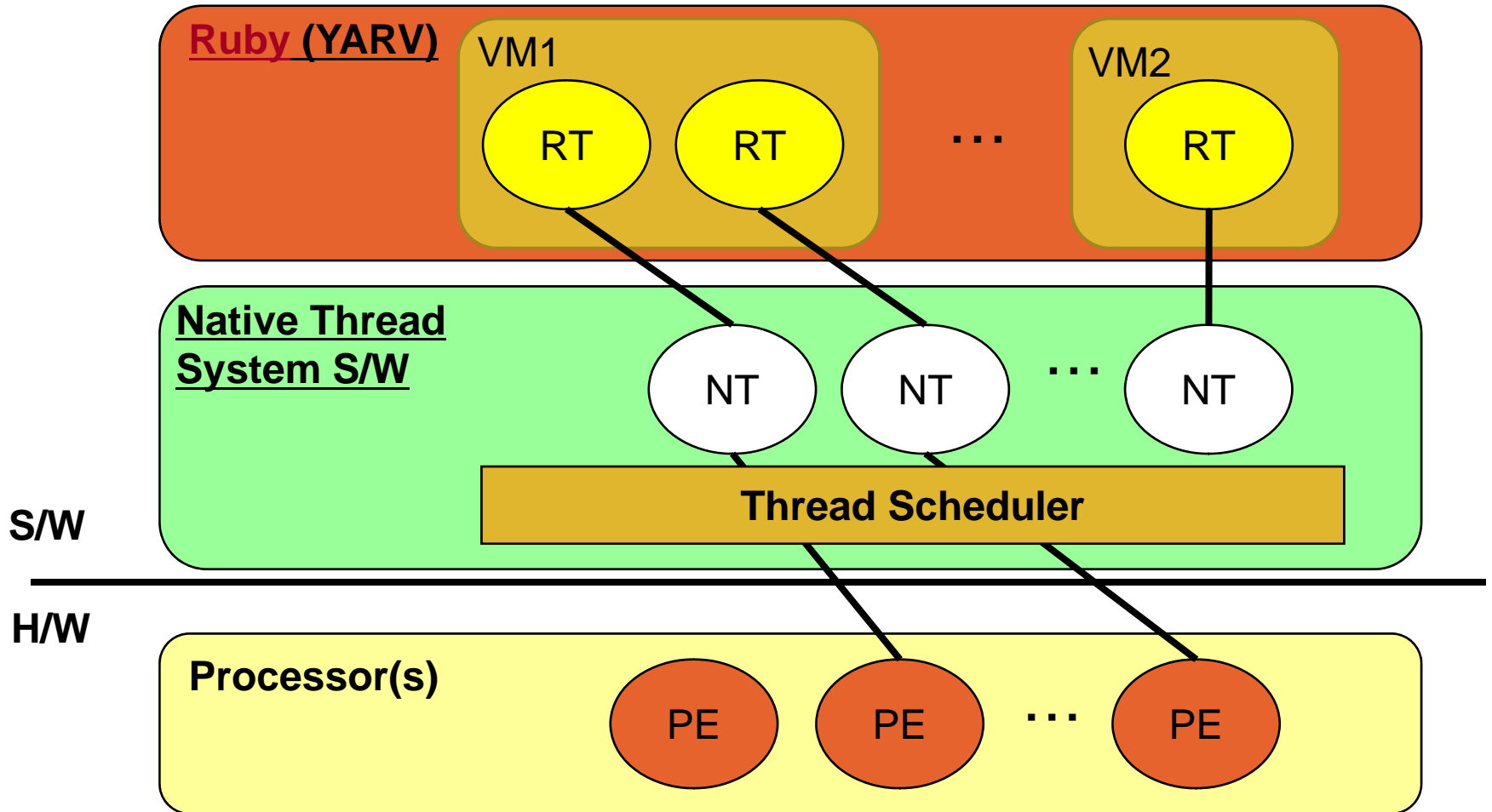
By The Way, Other CRuby GC Related Projects

- Generational GC (Kiyama)
- 1 bit Reference Count GC (Matz)
- Floating as Special Constant (ko1)
- Lazy Sweep (autherNari)
- Bitmap GC (Enterprise Ruby, autherNari)
- Mostly Copying GC (Ugawa)

Multi-VM (MVM) Project

- Multi Virtual Machine in One Process
- Each VMs are able to run in Parallel
 - Each VMs have Giant VM Lock.
- High Speed Inter-VM Communication
 - Inner Process Communication

Multi-VM Overview



PE: Processor Element, UL: User Level, KL: Kernel Level

Multi-VM (MVM) Project

Sponsored by
Sun Microsystems, Inc.

Nobu (a.k.a Patch Monster)
is Working for This Project

MVM

svn co

[http://svn.ruby-lang.org/
repos/ruby/branches/mvm](http://svn.ruby-lang.org/repos/ruby/branches/mvm)

Summary

CRuby/YARV is
NOT “BEST” Solution
for Performance.

However, CRuby/YARV is
“GOOD” Solution for Us,
the Pragmatic Ruby Programmers,
at least Several Years.

Summary (cont.)

- CRuby is Enable to Evolve Moreover
- Some Projects to Take advantage of CRuby
 - Ricsin: mix-in C to Ruby Project
 - Ruby to C AOT Compiler Project
 - atomic-Ruby Project
 - Multi-VM Project

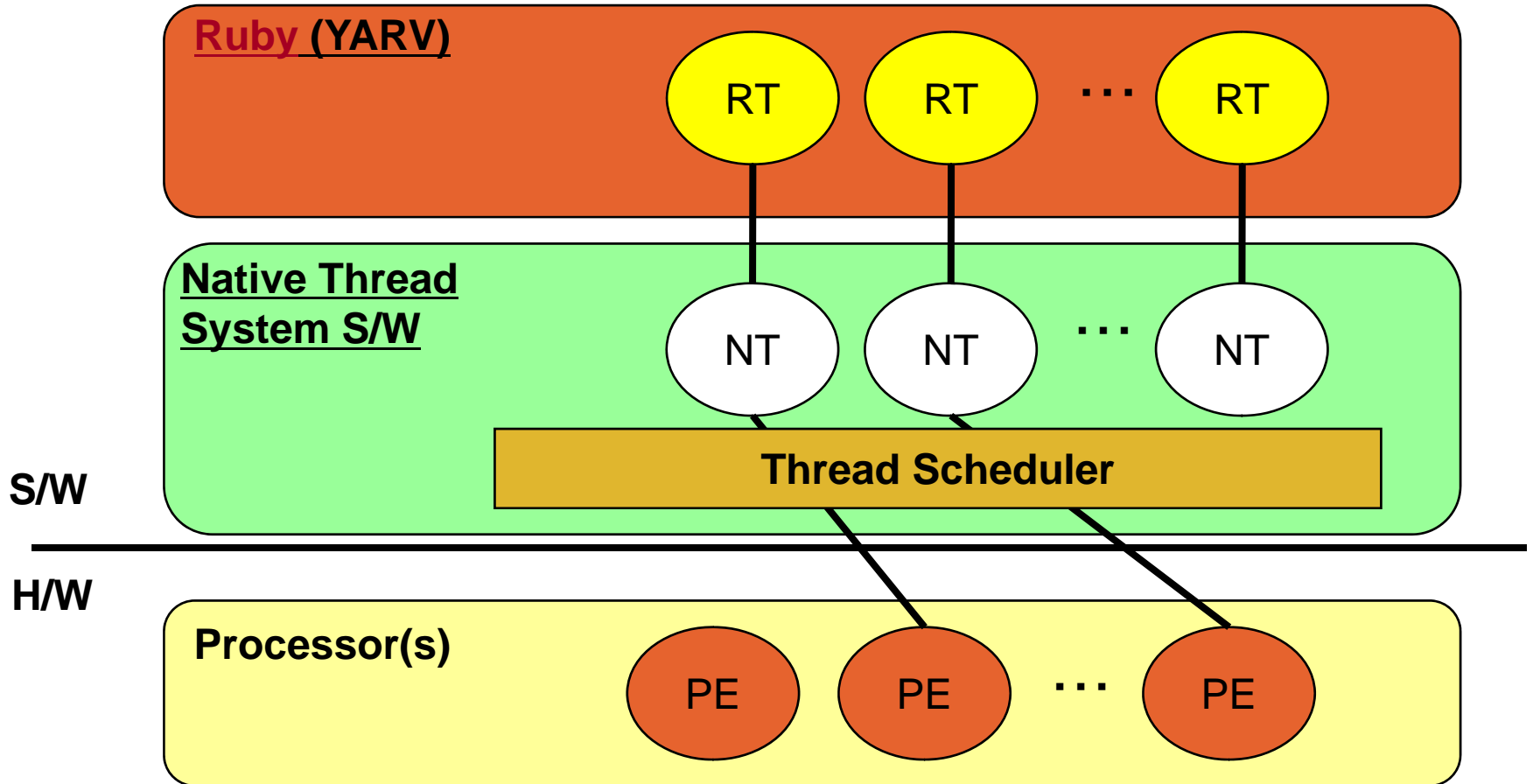
Fin.

Thank You for Your Attention.
Any Questions?

SASADA Koichi
<ko1@rvm.jp>

Department of Creative Informatics,
Graduate School of Science and Technology,
The University of Tokyo

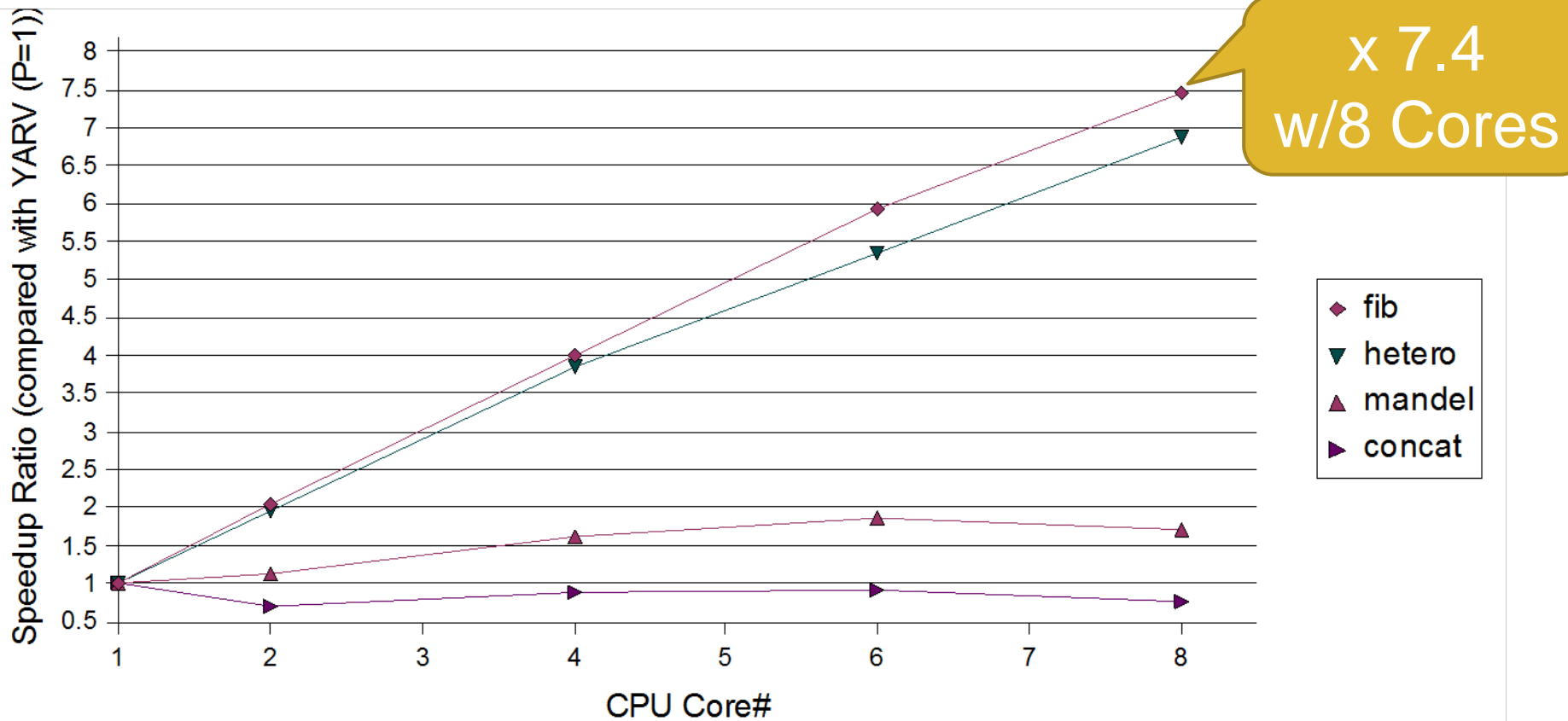
Accepted Method: Ruby Thread and Native Thread (1:1) ← Ruby 1.9/YARV



— PE: Processor Element, UL: User Level, KL: Kernel Level



Evaluation Result (Micro-benchmark)

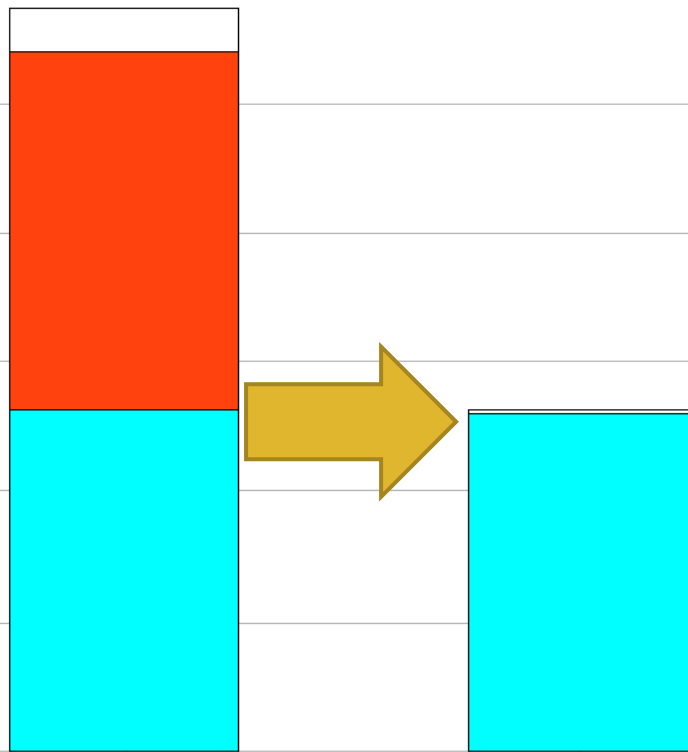


Discussion

How to Embed 64 bit Double?

- VALUE embed Object doesn't need memory overhead
- **64bit CPU have 64 bit pointer type**
→ **Use 64 bit CPU**
- At least we need 1 bit for TAG bit
 - From Mantissa?
 - Decrease Precision
 - From Exponential?
 - Decrease Representation Range

Evaluation Toy-Program



- Others
- Mem
- Calc

- ✓ Reduce Mem Time
- ✓ Encode/Decode don't affect to Performance

Evaluation

Compared with other Ruby Impl.

