

Implementation of Ruby 1.9.3 and later

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



Today's topic

- Ruby 1.9.3
- Next and next version of MRI

We released!

Ruby 1.9.3

...rc1

Yugui-san (release manager) says ...

“I will release Ruby 1.9.3 within this two weeks unless any serious problem is reported. If you have any trouble with Ruby 1.9.3, please let us know.”

It's means...

We need human sacrifice



<http://www.flickr.com/photos/babomike/4741964697/>

Ruby 1.9.3 New Features

RTFN

Read the F*****g News

Ruby 1.9.3 Features

- License
 - Original and (GPLv2 → 2-clause BSD)
- Syntax, Methods/Libraries
 - Private constants
 - DL/YAML → Fiddle/Psych
 - Test::Unit → Parallel extension (by @sora_h, talking at next room)
 - String#prepend, IO.write, etc
 - ... (Actually, I don't know about Ruby world)



Implementation of Ruby 1.9.3 Internals and later

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



Ruby 1.9.3

Interpreter internals

- Garbage Collection
 - Lazy Sweep
 - Parameter tuning
- Timer thread implementation
 - Reducing power consumption
- GVL implementation for multi-core
 - Solve an problem about non-thread switching on multi-core

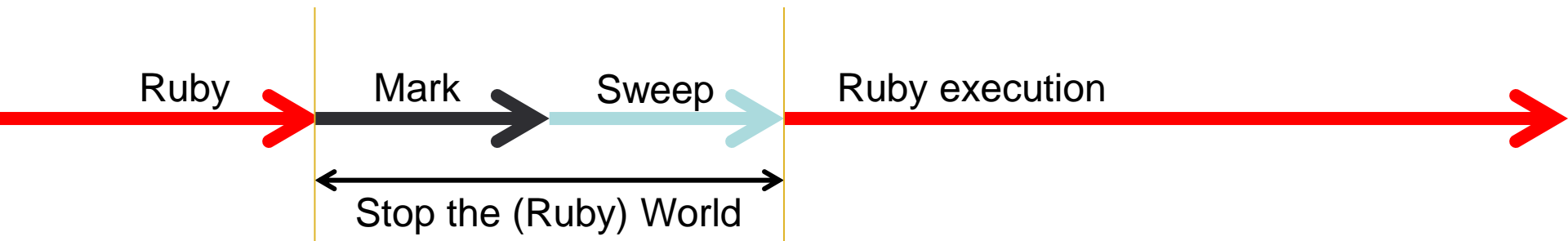
Garbage Collection



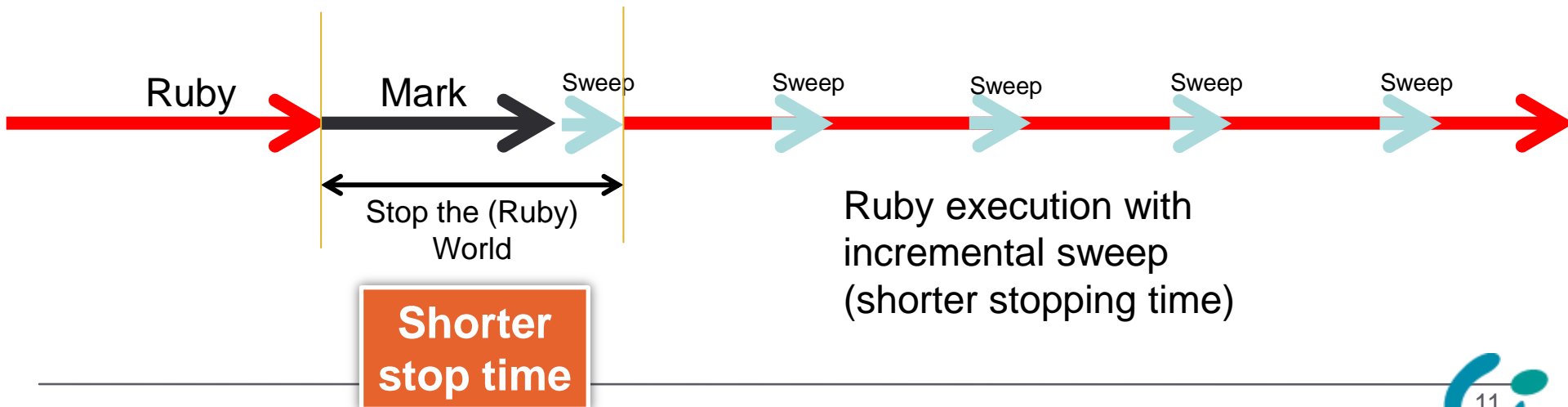
<http://www.flickr.com/photos/ninithedreamer/4649075746/>

Introducing Lazy Sweep GC

- **Before 1.9.3:** Stop the world mark and sweep



- **After 1.9.3:** Stop the world mark, and incremental sweep



Garbage collection Parameter tuning

- ✓ You can tune GC by using environment variables.
- ✓ RUBY_GC_MALLOC_LIMIT
- ✓ RUBY_HEAP_MIN_SLOTS
- ✓ RUBY_FREE_MIN

Read a source code for details :p
Quoted from nari3's slide

Recycle Presentation
@RubyKaigi2011, again

Ruby 1.9.3

All About Timer Thread (core, details)

Koichi Sasada

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

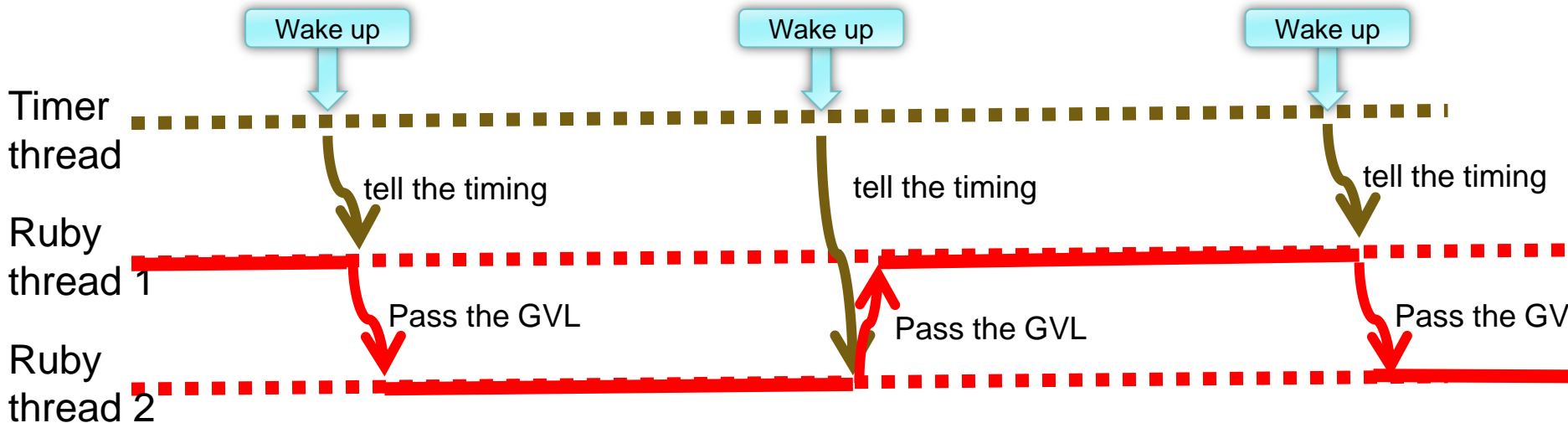


Power consumption



Background Timer Thread?

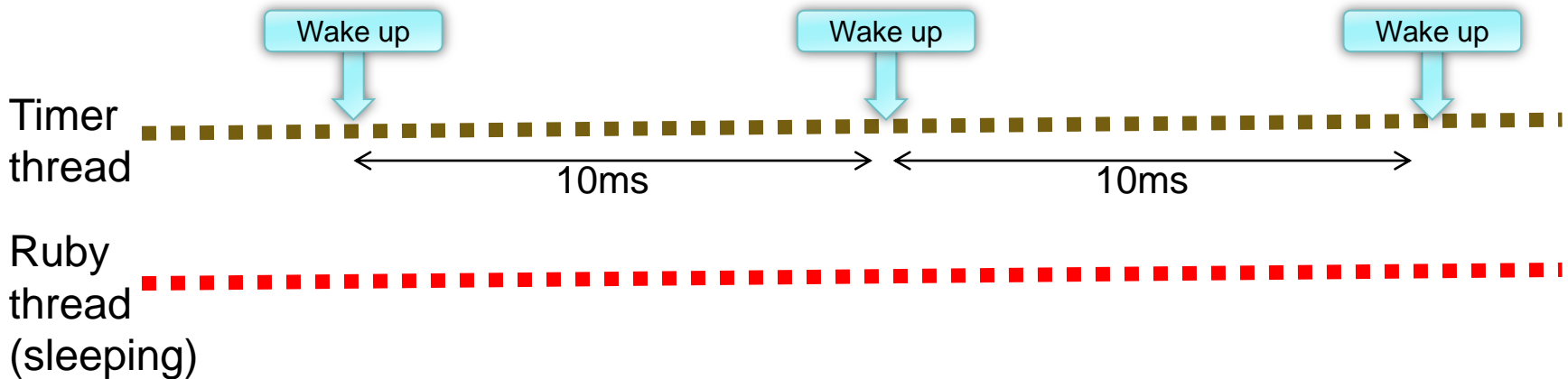
- Timer thread is an internal mechanism on Ruby 1.9
 - For thread scheduling
 - For signal delivery



Problem

Power Consumption?

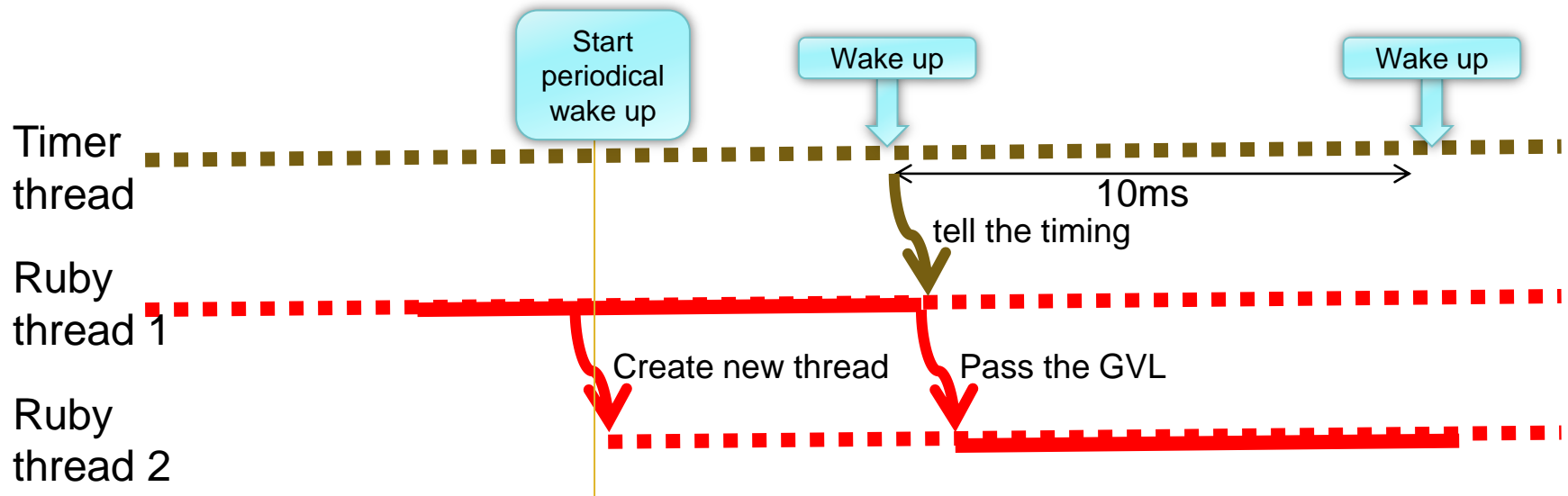
- Timer thread awake every 10ms
 - To notify a running thread to release GVL
 - Wake-up anytime even if it is not necessary (no scheduling is needed) → CPU can't sleep enough
 - A few people claimed that the timer thread consumes power



Solution

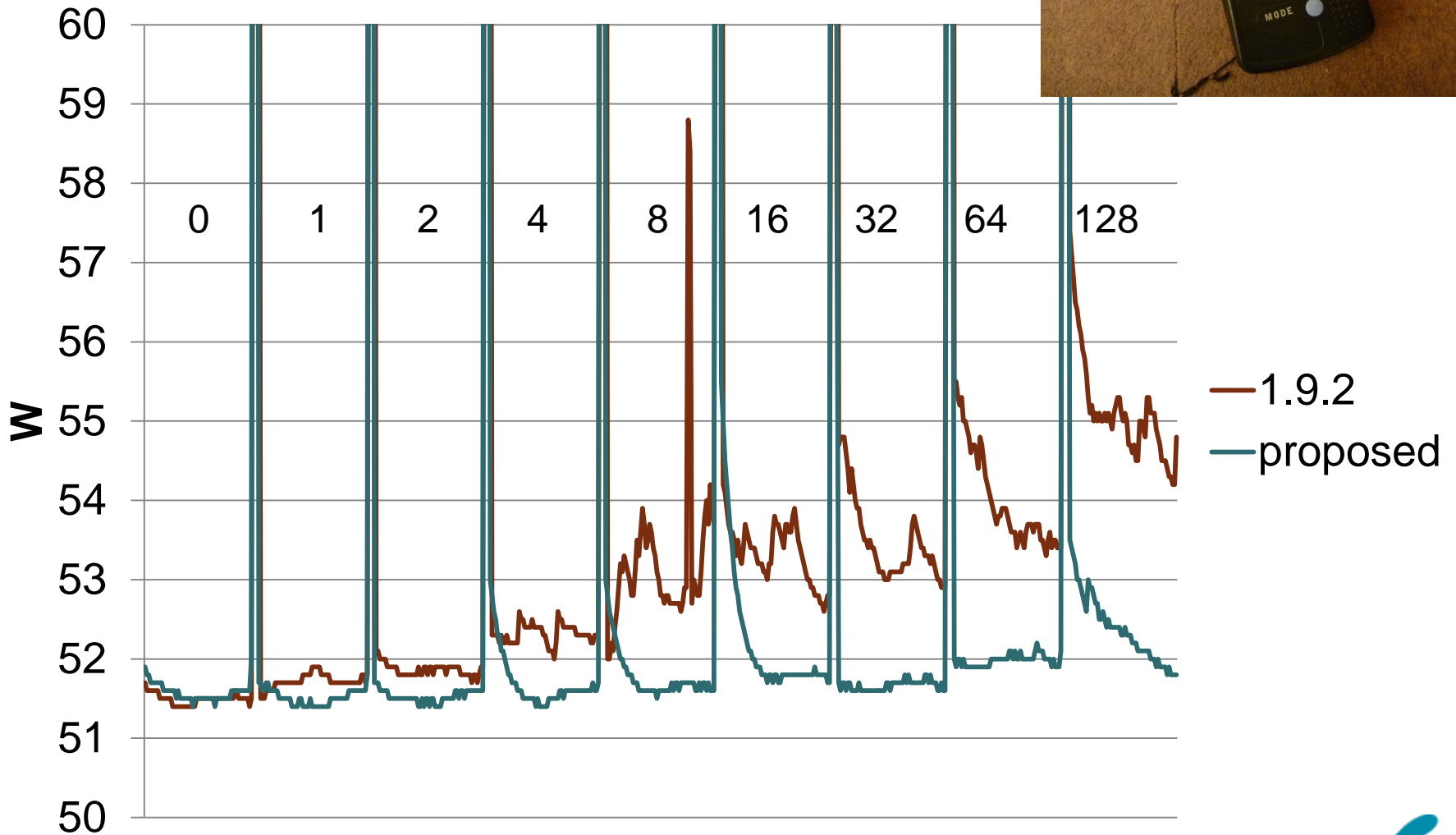
Rewrite Timer Thread Code

- Make timer thread more smart
 - Timer thread **sleeps infinitely** if only one thread is needed
 - Using **pipe trick**

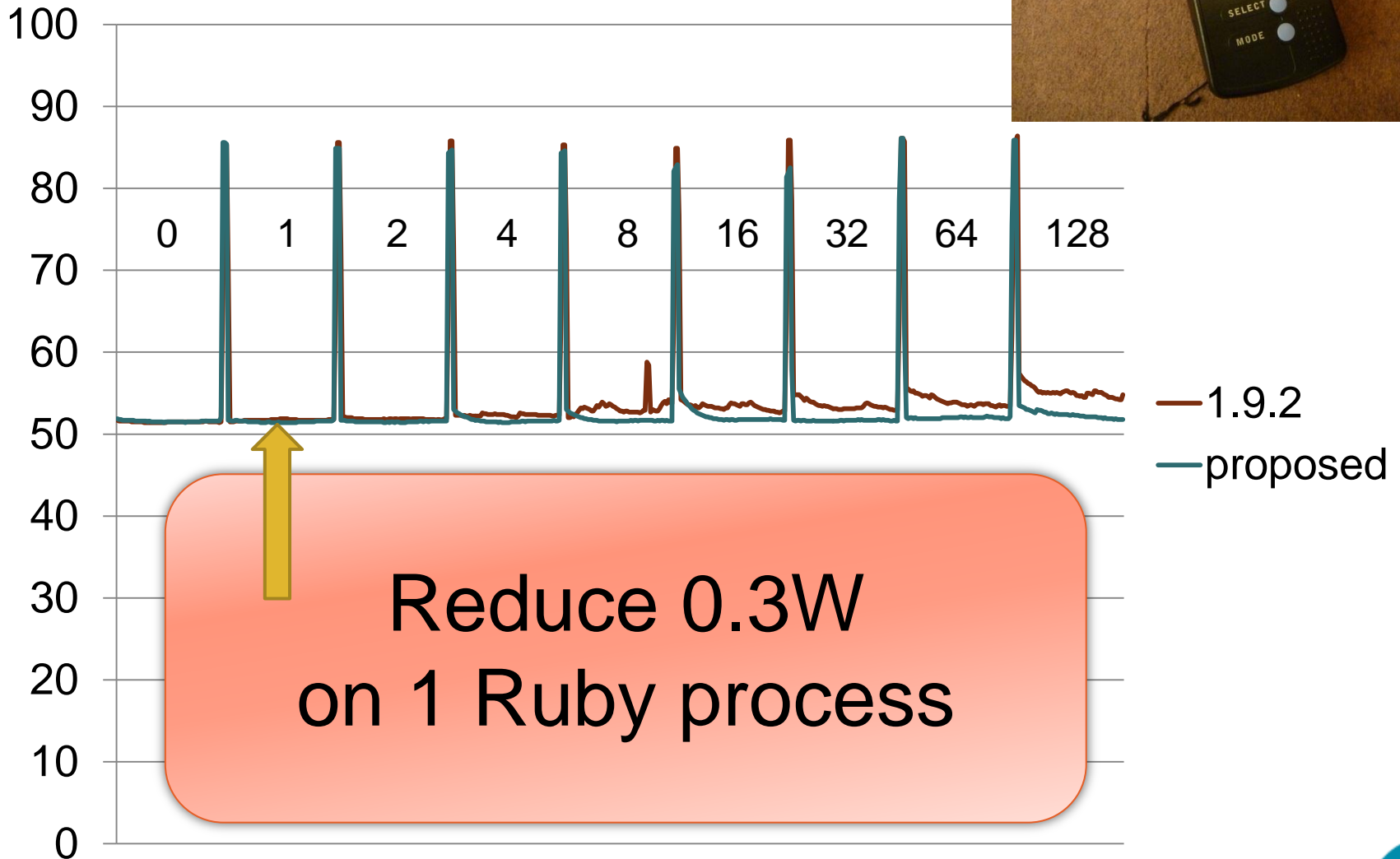


Evaluation (?)

Power consumption



Evaluation

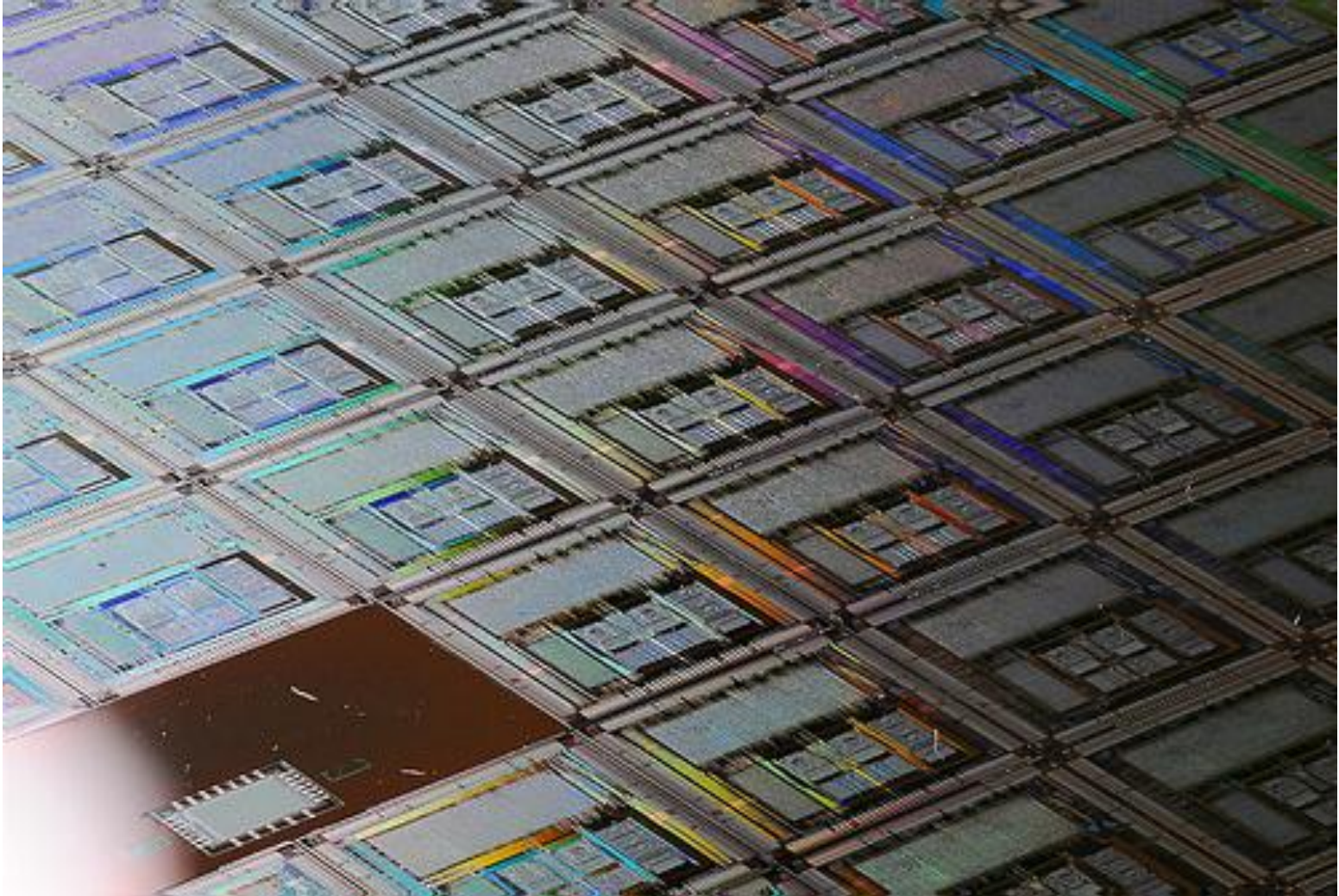


Caution!

Incompatibility

- Pipe trick needs additional file descriptors
 - You should not close them
 - **Passenger does it!!**
- Skip to close such file descriptors w/ new C API “**rb_reserved_fd_p()**”

Thread Scheduler on the multi-core



Recycle Presentation
@RubyKaigi2011 by Kosaki-san

Ruby 1.9.3

GVLおよびロックの改善

GVL and Lock improvement

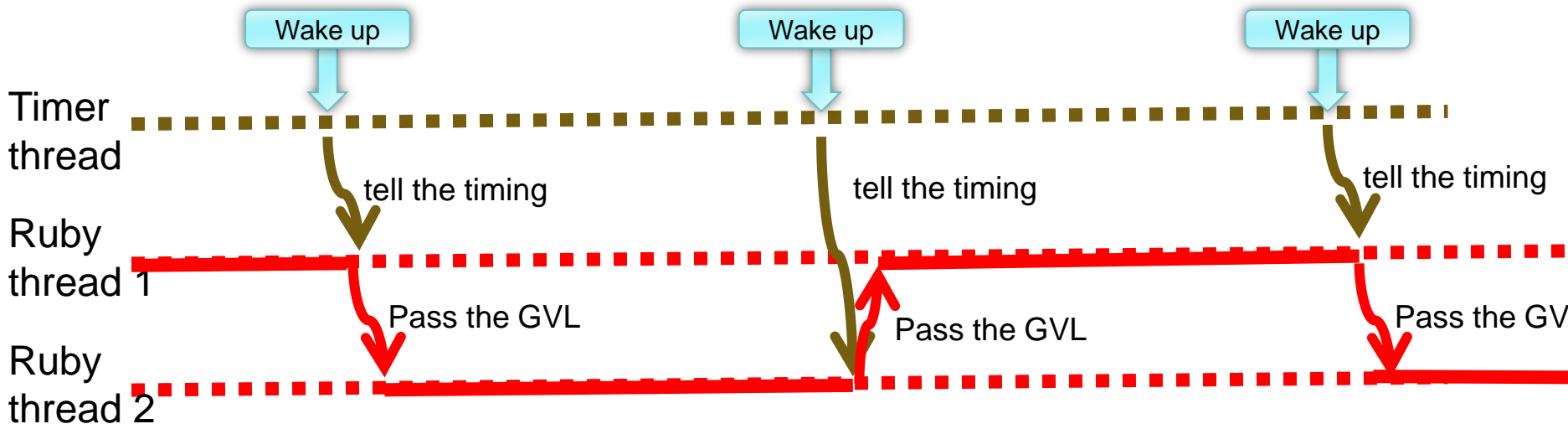
He is a
Ruby and
Linux kernel
committer

Fujitsu
Motohiro Kosaki

Background again

Only one thread can run w/GVL

- Timer thread is an internal mechanism on Ruby 1.9
 - For thread scheduling
 - For signal delivery

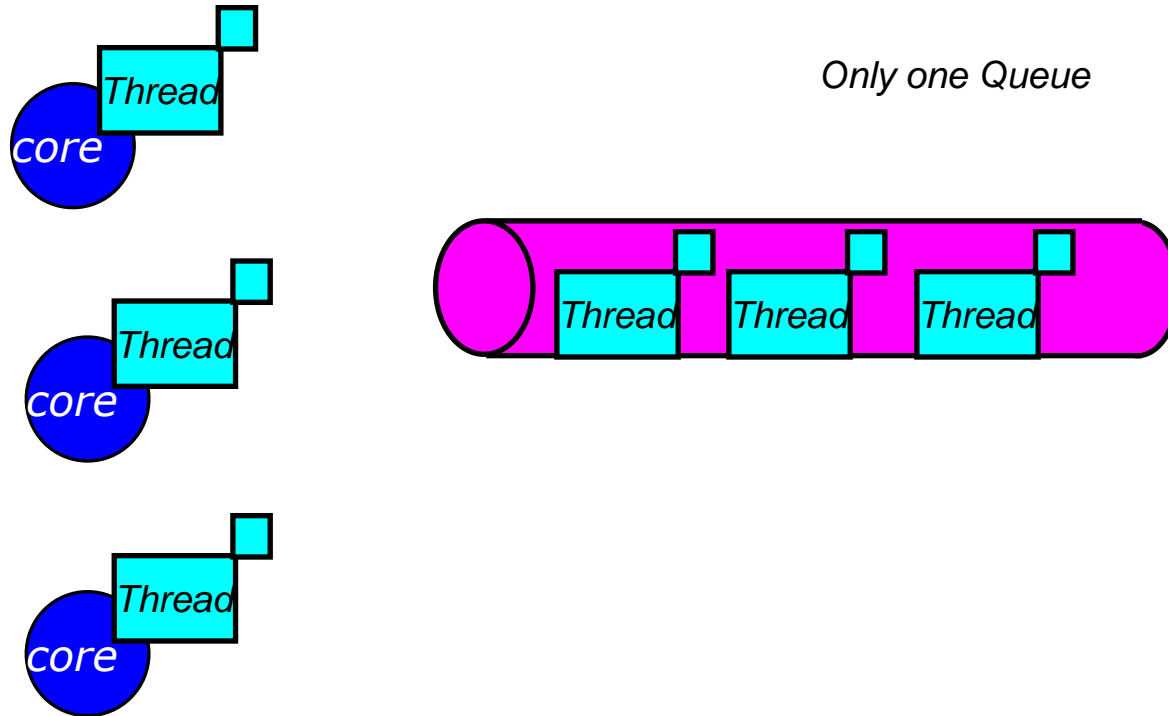


Doesn't work some program on Multi-core environment

```
# example
f = false
Thread.new {
  ....;
  f = true
}
Thread.pass until f
```

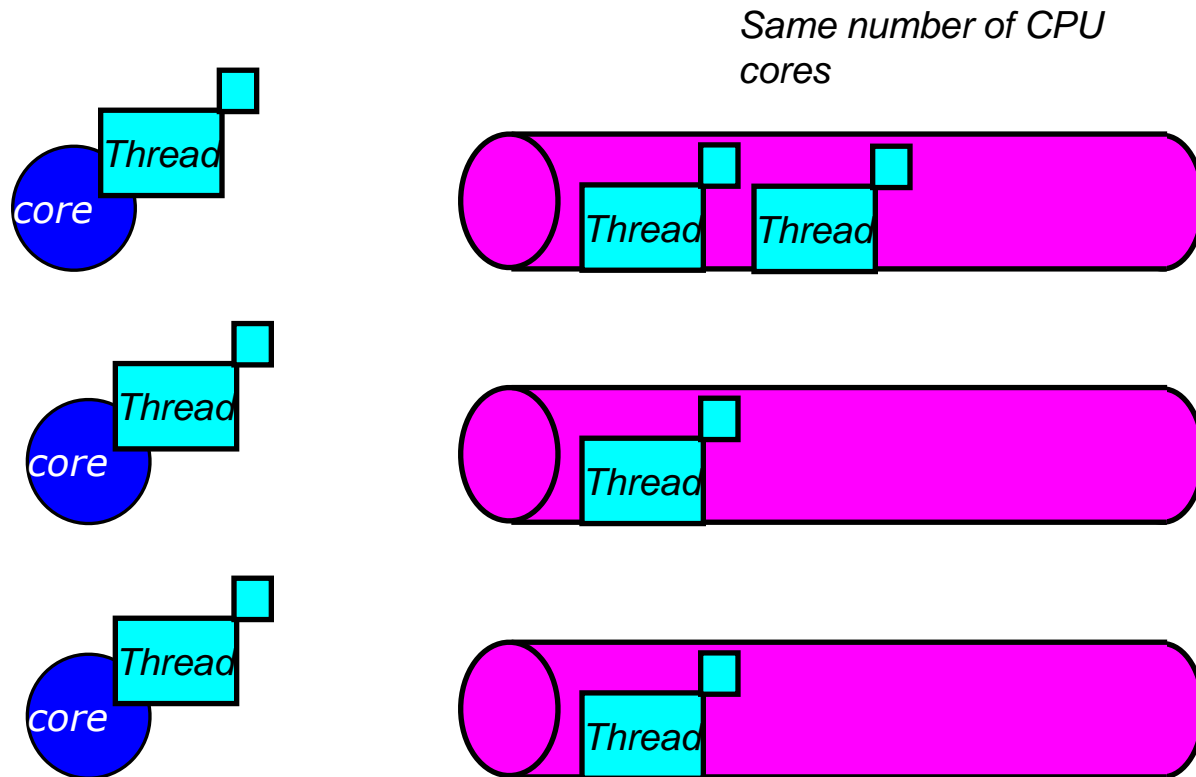

Why?

Good old OS scheduler



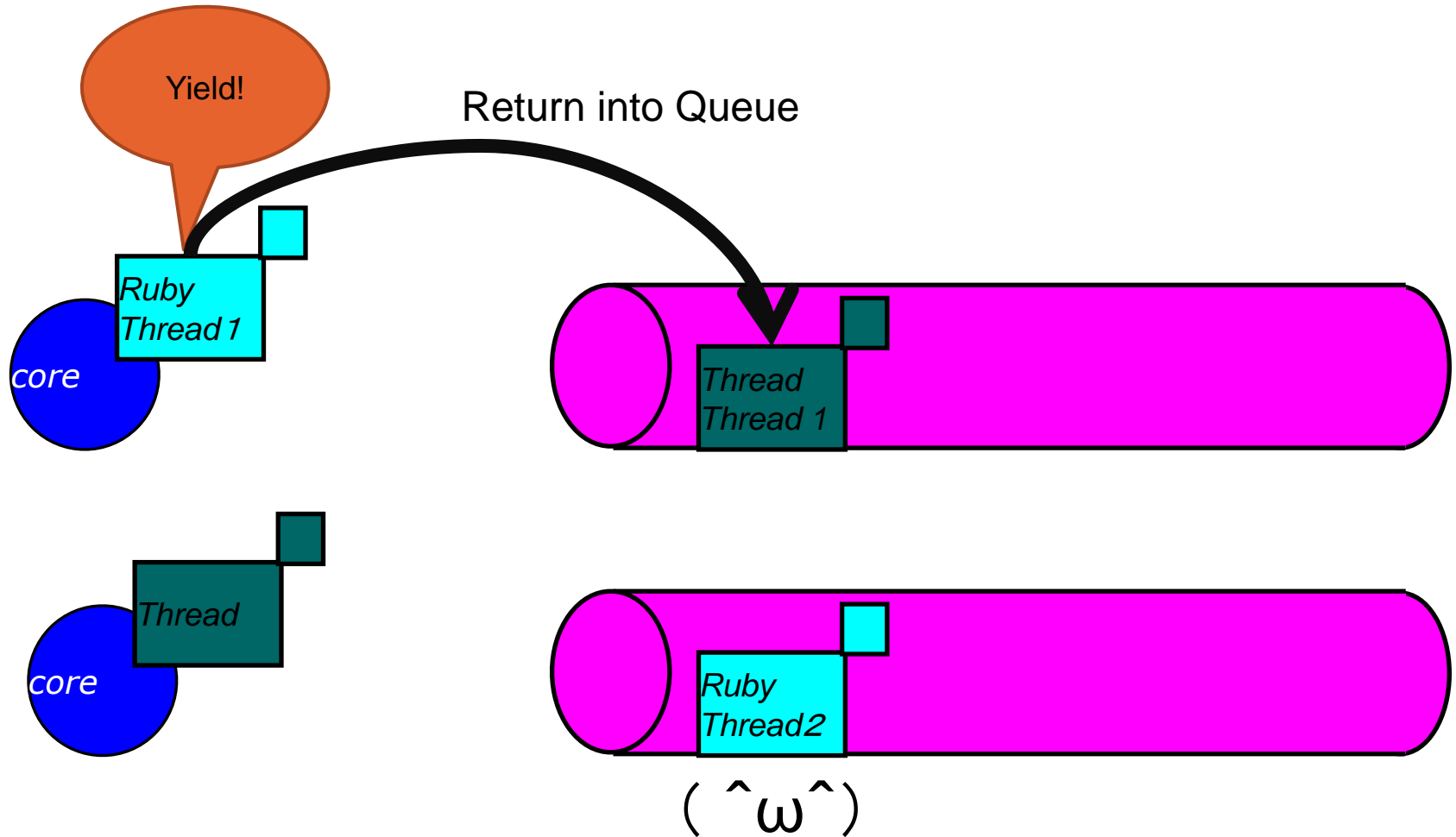
Quoted from Kosaki-san's slide

New fast OS scheduler



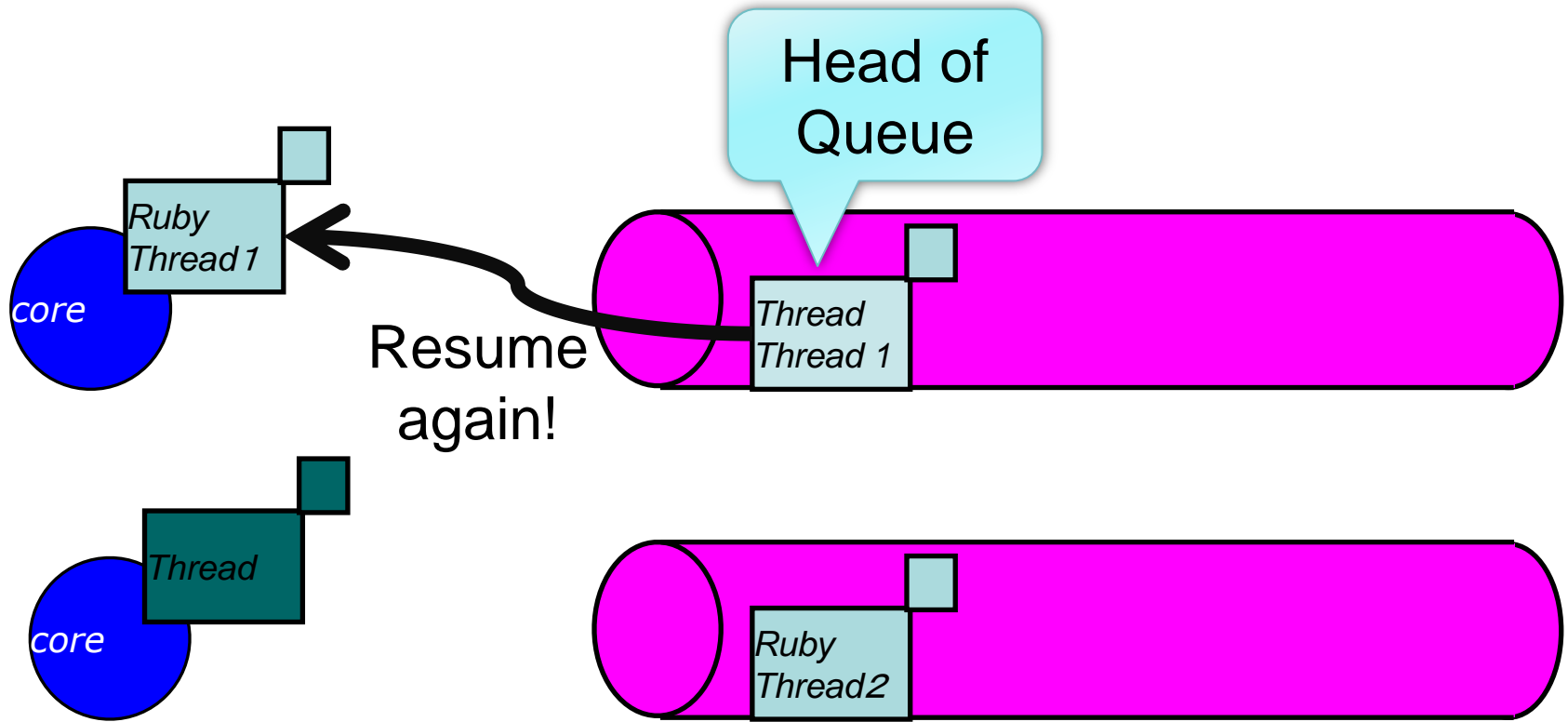
Quoted from Kosaki-san's slide

Situation (1)



I can expect resume!

Situation (2)



(´・ω・`)

I can't resume again...

Solution

- We change GVL passing strategy
 - Passing GVL definitely
 - Ask me later if you want to know

gvl_acquire

```
mutex_lock(&lock->lock)
cond_wait(&lock->wait)
lock->acquired = 1;

// notice
cond_signal(&lock->switch_cond)

mutex_unlock(&lock->lock)
```

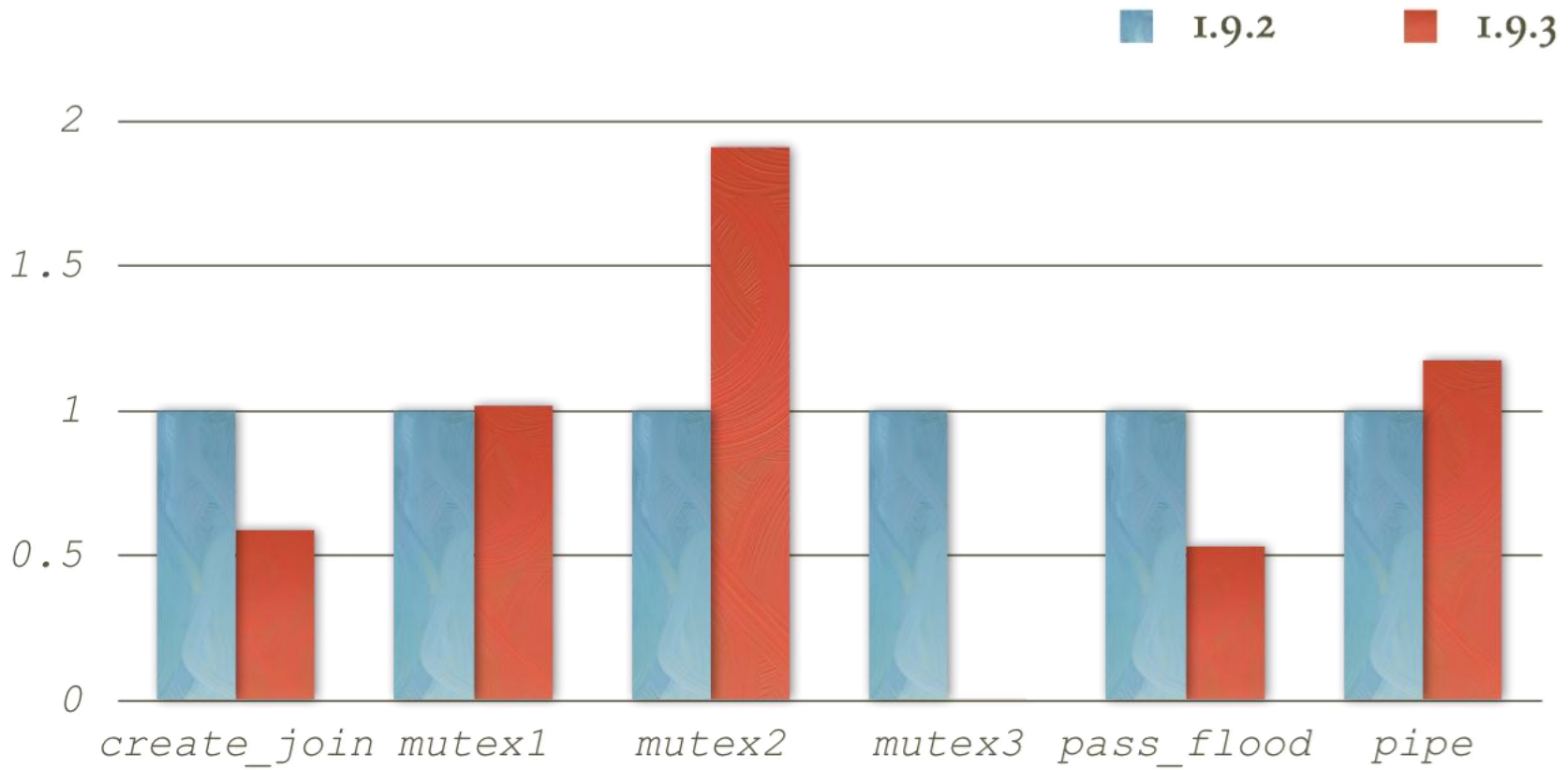
gvl_release

```
mutex_lock(&lock->lock)
lock->acquired = 0;
cond_signal(&lock->wait_cond)

// sleeping until
cond_wait(&lock->switch_cond)

mutex_unlock(&lock->lock)
```

Thread related performance



Future



<http://www.flickr.com/photos/viernullvier/5698630227/>

Implementation of Ruby 1.9.3 and later

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



Ruby 1.9.4? Ruby 2.0?

- Now discussing which version should be released
 - Matz says “Next version is 2.0 including several new syntax”
 - keyword argument support for method definitions
 - `Module#mix`
 - `Module#prepend`
 - and others (refinement, classbox, or method shelter?)
 - Another says “Should be 1.9.4”

Ruby 2.0?

What feature do you want?

**Teach me here,
raise your hand!
Anything is okay.**



OKay.

There are many
requirements.

Ask Matz

Yesterday, I asked Matz what should we do. He doesn't want to release 1.9.4 anymore. He want to say there are no progress in 1.9 series any more, but only in 2.0 series. If spec is concluded, he shrink 2.0 spec and release force.

Please ask Matz tomorrow,
if you have opinion about it.

My concerns on next version

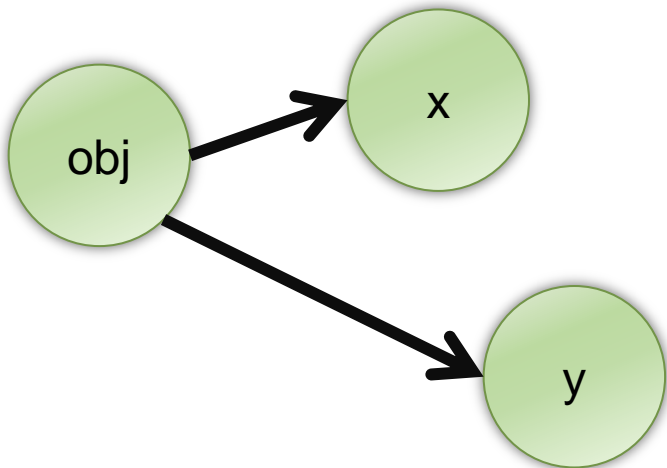
- Performance
 - This is why I'm here
- Profiler/debugger interface
 - Please teach me what information you want to know!
- Compiling supports
- I'll introduce several possibility and activities about "Ruby in Future"

Profiling / Debugging support

- We add more support for profiling/debugging
- Please teach me what information you want to know!

ObjectSpace.reference_from(obj)

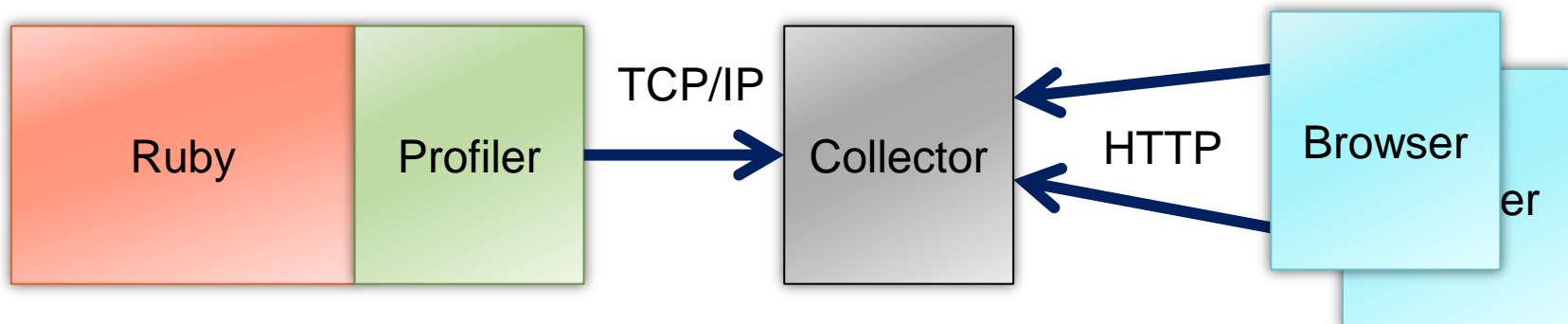
- 2 days ago, I posted a patch of `ObjectSpace.reference_from(obj)`
 - I'm sad there are few feedback...



```
ObjectSpace.reference_from(obj)
#=>
{98752943875 => x,
 48334232r23 => y}
# keys are object id
# value are object themselves
```


Research of my student: Performance profiler

- ll-prof
 - Real-time profiler for LL languages
 - Ruby and Python (he is Pythonian)
 - Viewer in JavaScript
 - You can see results in you browser



Demo

(movie made by student)

Performance

- Speed
 - VM w/ runtime performance
 - GC performance
- Memory consumption
- Power consumption

Performance VM w/ runtime speed

- Performance improvements VM
 - Easy escape analysis to prevent generating temporary objects

Performance

VM w/ runtime speed

- Semi-automatic Type inference and translate Ruby code into C (C extension)

Student's research: CastOff

A performance improvement tool for ruby1.9.3

(1) Use from
ruby script



Programmer

```
require 'cast_off'  
CastOff.compile(  
  Klass,  
  :Method,  
  binding,  
  TypeInfo)  
...
```

- Compile
Klass#Method
- Load compiled binary
- Run faster

(2) Use from
command line



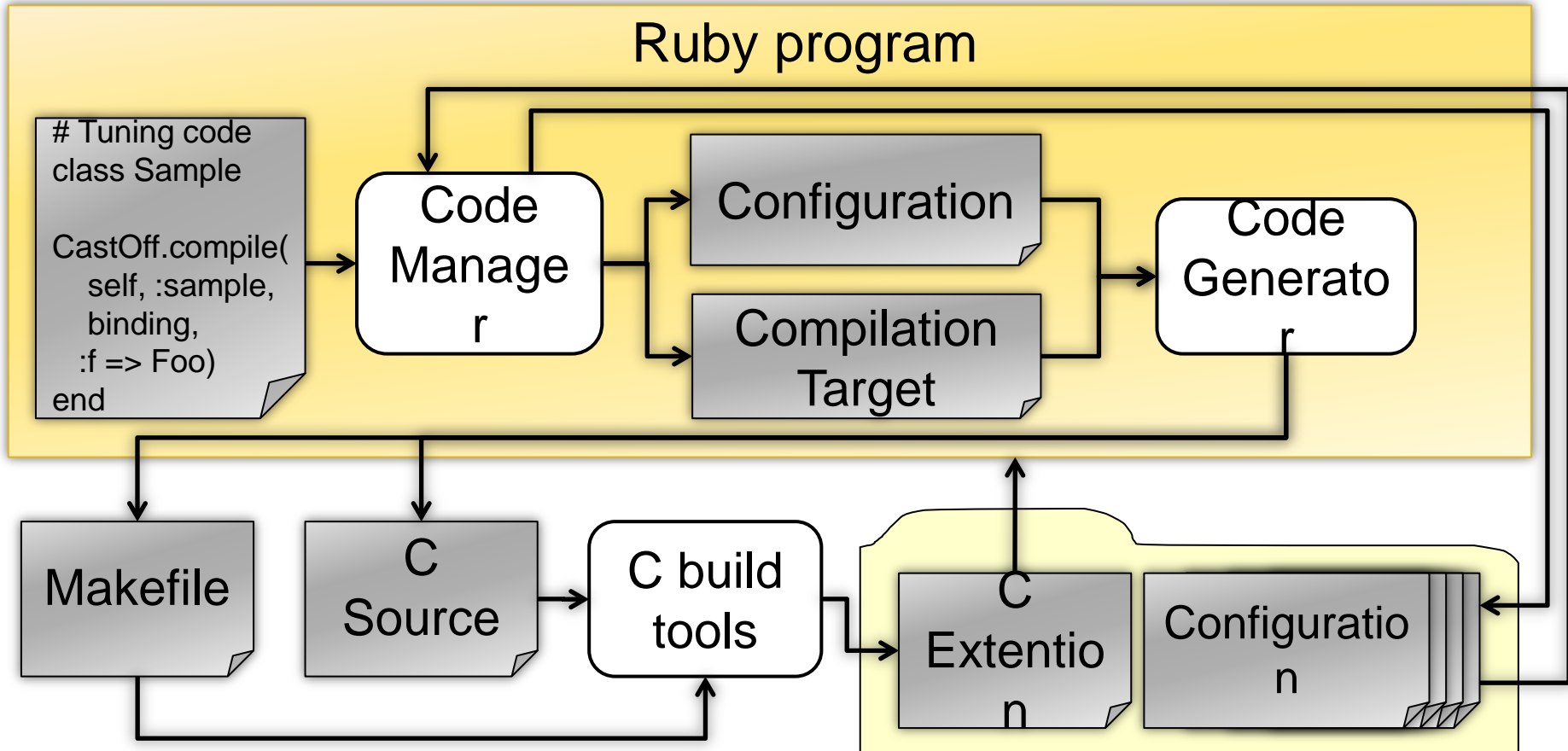
Programmer

```
$ CastOff  
"program"
```

- Run and Profile
"program"
- Compile methods
in "program"
- Run faster

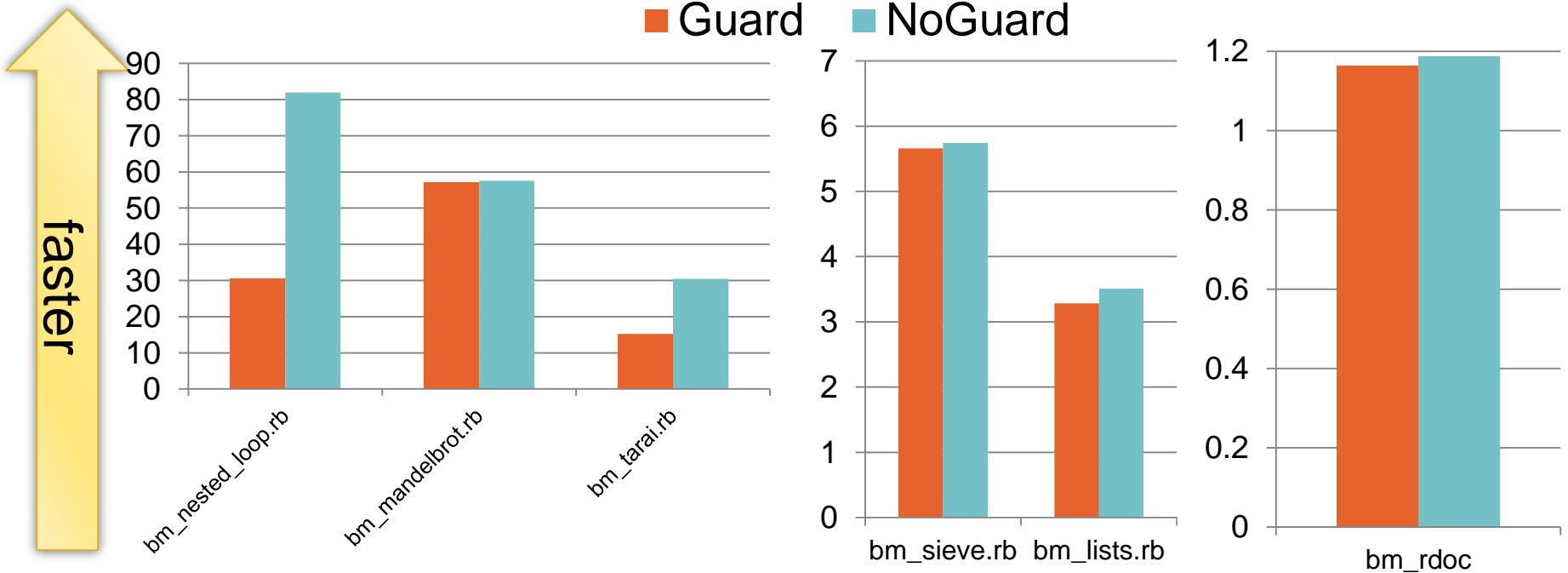


Compilation flow



Performance improvements

Execution time ratio (CRuby 1.9.3 / CastOff)



Performance

Garbage collection

- MRI/CRuby's GC is slow
 - Stop the world and mark all
- Generational?
 - Issue: C extension compatibility
- Ideas
 - Parallel GC → Yesterday's nari3's talk 😊
 - Escape analysis and reduce GC managed objects
 - Smart data-structure to reduce GC managed objects

Performance

Memory consumption

- One Rails app consume 100MB
- Encourage sharing resources
- such as Strings, Arrays, and so on
 - REE/Kiji CoW friendly, generational

Performance

Power consumption

- Time thread modification at 1.9.3 → small one
- More aggressive way?

Performance Parallel computing



Parallel Execution

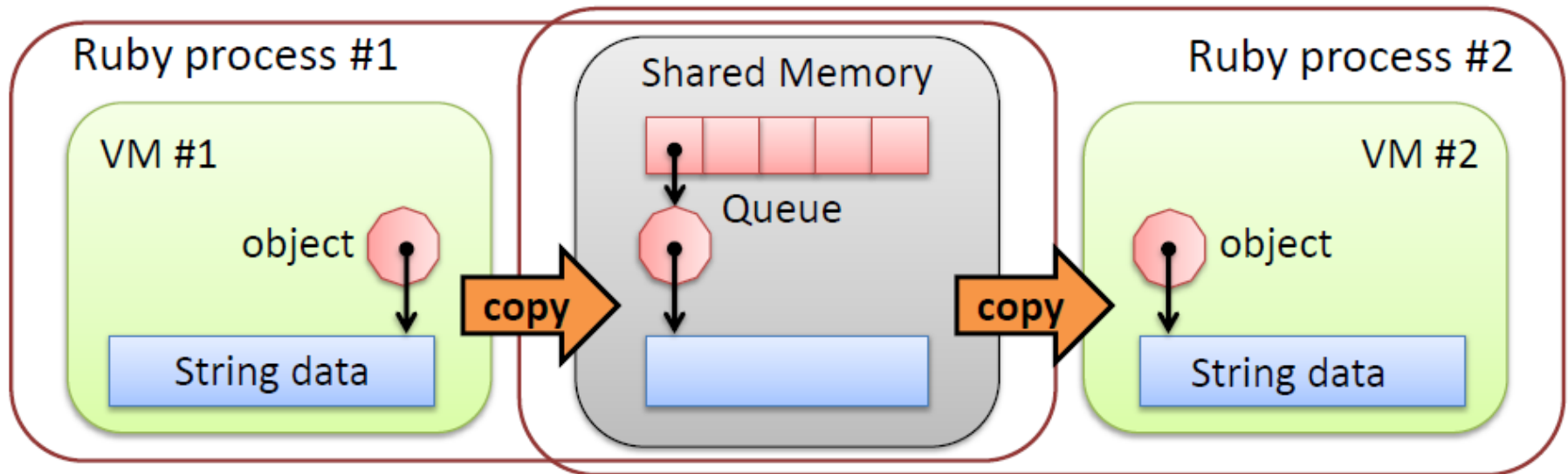
- Run threads in parallel (JRuby, MacRuby, ...)
 - Good: Well known approach
 - **Bad: Difficult to make safe/correct multi-threaded programs**
 - Many tragedy (in Java, etc.)
 - Bad: Difficult to make efficient implementation with fine-grain lock
- Parallel processes (dRuby, ...)
 - Good: No need to implement
 - Bad: Marshal overheads

Support friendly Coarse-grained parallel computing

- Encourage Multi-process
 - Traditional well-known approach
 - Toward advanced dRuby
- Multi-VM
 - VMs in one process
 - Light-weight communication

Student's research Tunnel: Inter-process communication w/shared memory

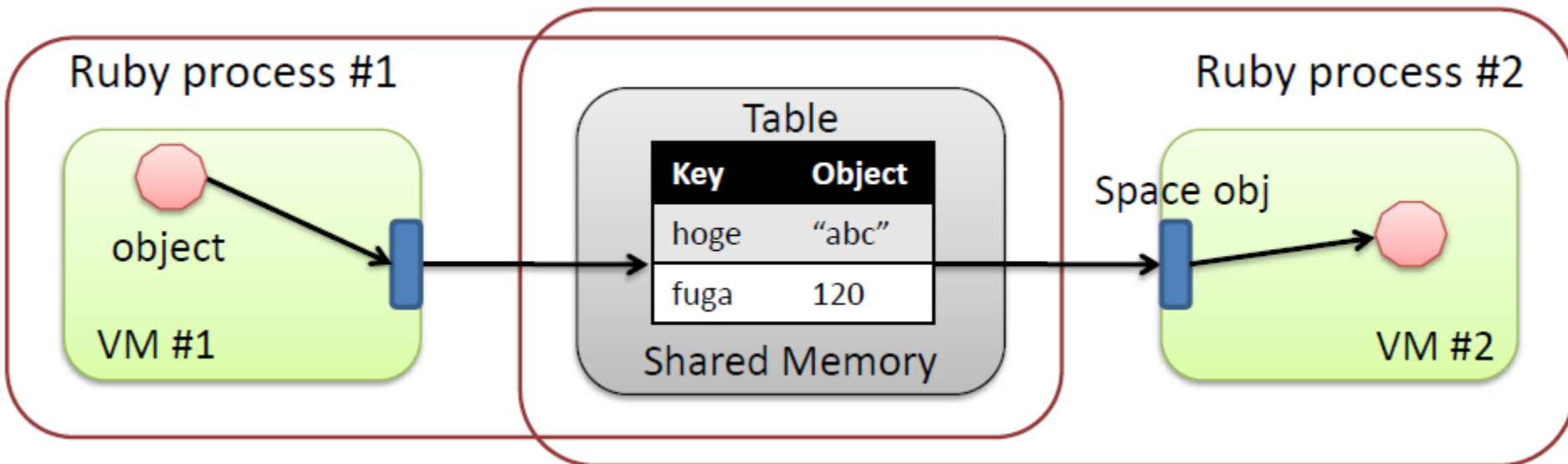
- Object transfer w/ shared memory



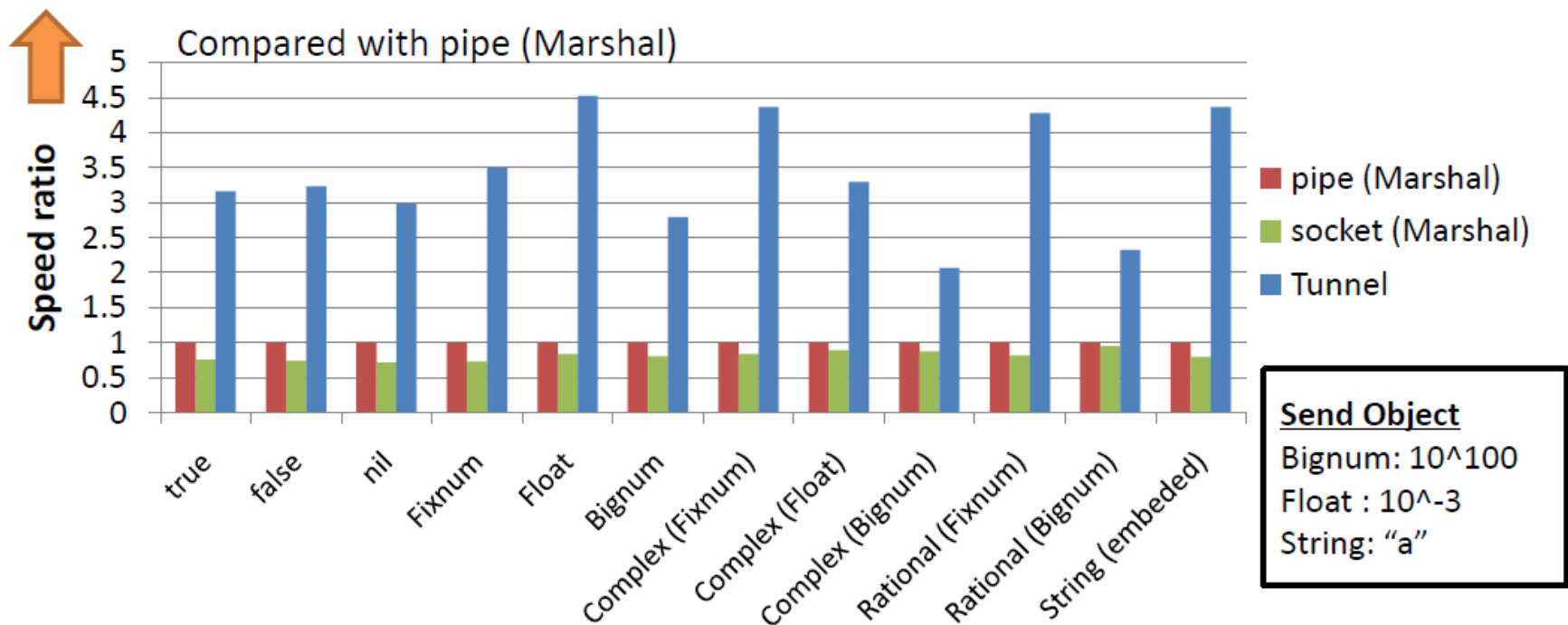
Student reserch (cont.)

Space: Inter-process Space w/shared memory

- Shared space between ruby processes
 - Similar to Linda/Rinda



Evaluation of Tunnel



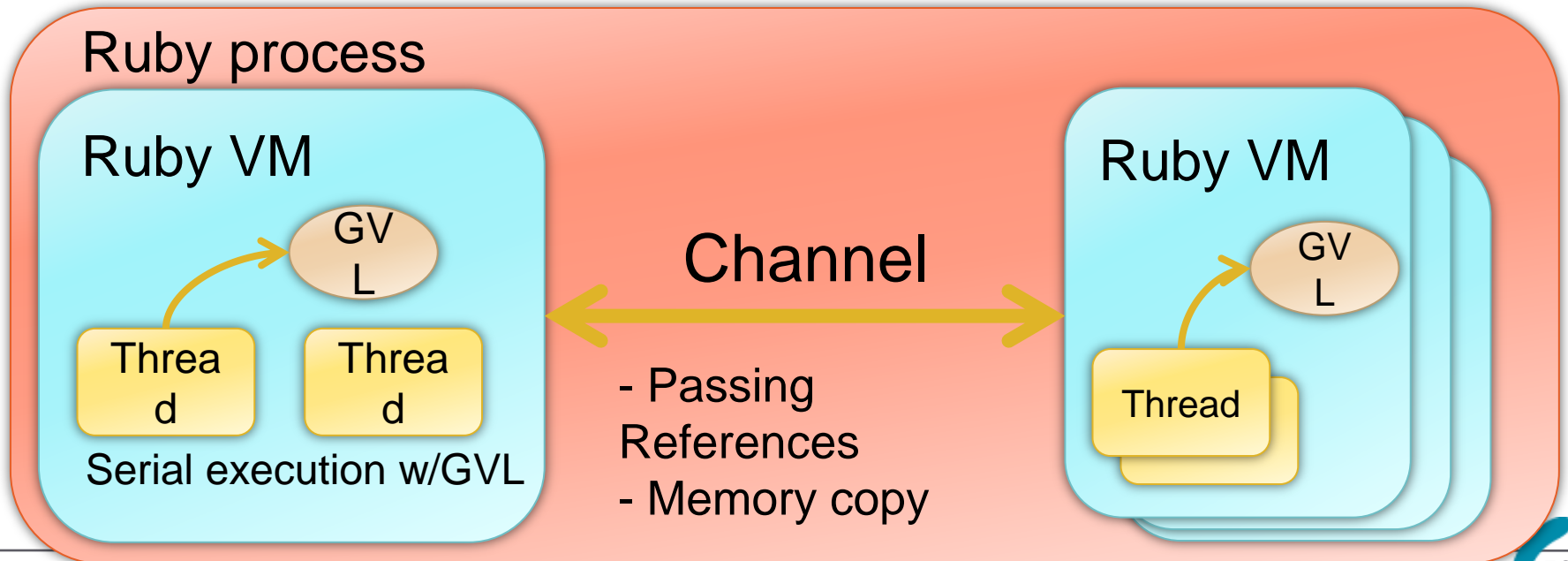
MVM

A.k.a. vaperware ☹️

We have progress on it, this year.

Parallel Execution Multiple-VM (MVM) on Ruby

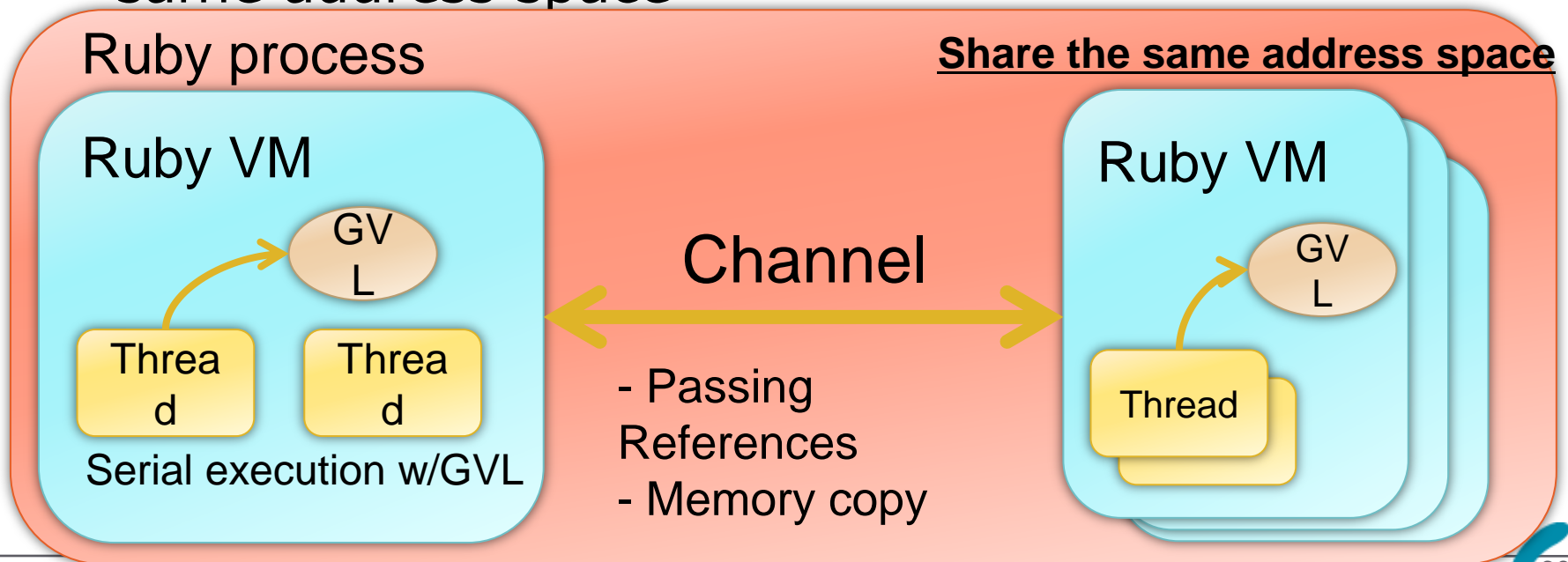
- Multiple VMs in one process
 - VMs are completely isolated (Each VM has an independent object space (heap))
 - VMs run in **parallel**
 - Each VM has own GVL (w/o fine grained lock)



Our Approach

Multiple-VM (MVM) on Ruby

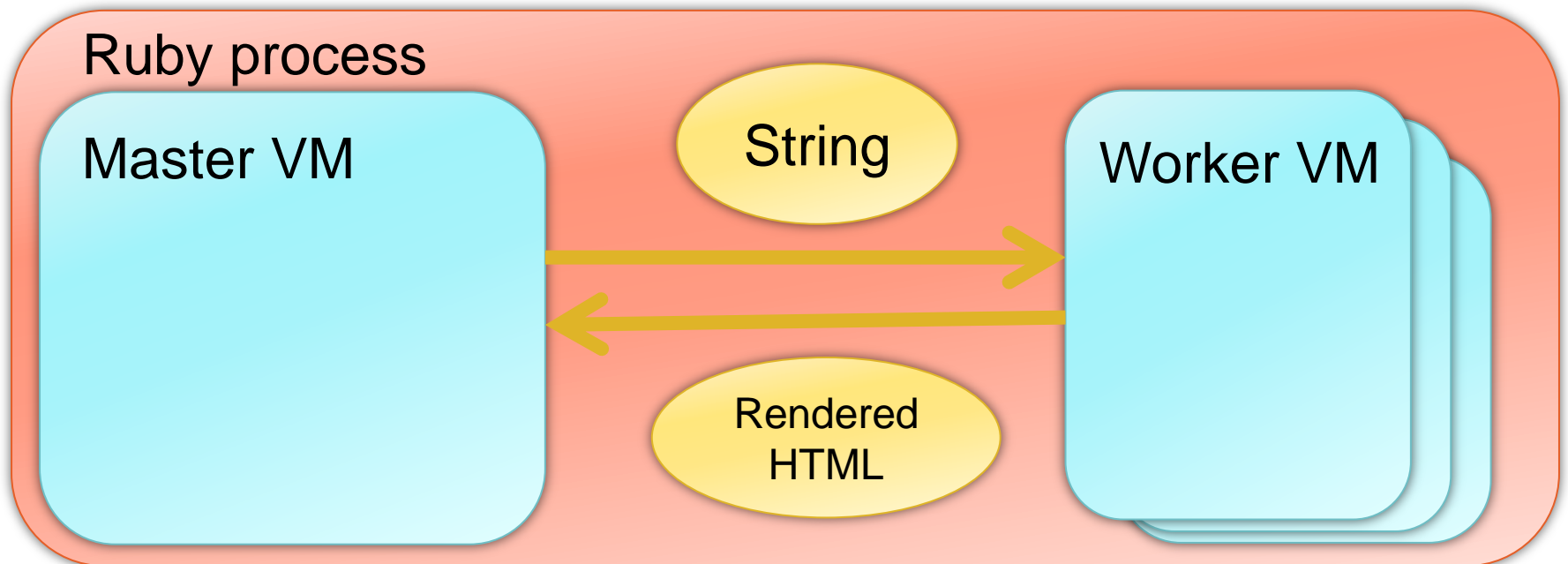
- Channel: Inter-VM Communication mechanism
 - The only way to communicate with other VMs
 - Simply passing references or copying memory in the same address space



Evaluation

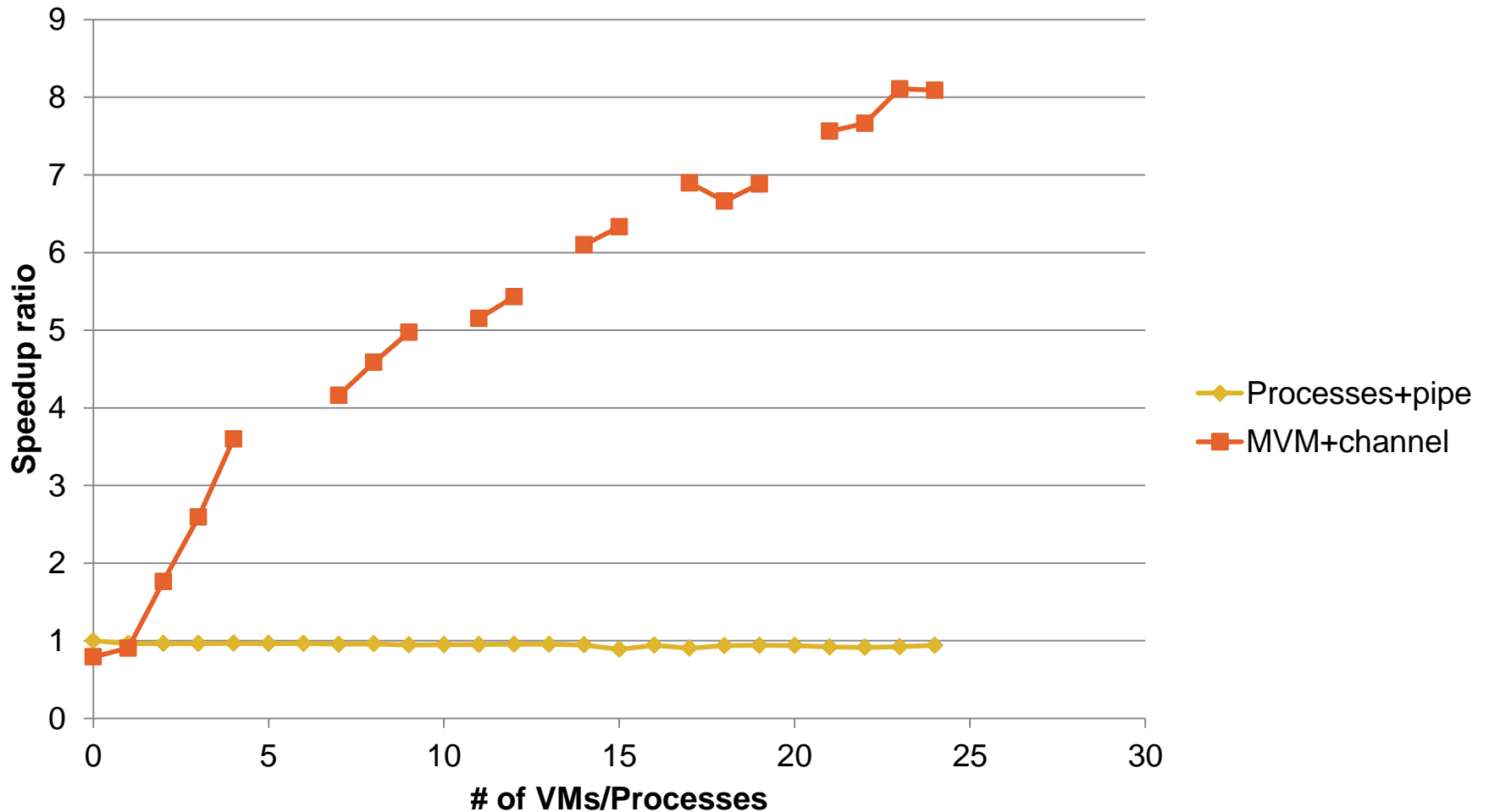
HTML rendering app

- Master dispatch string to worker and worker returns rendered HTML.



Evaluation

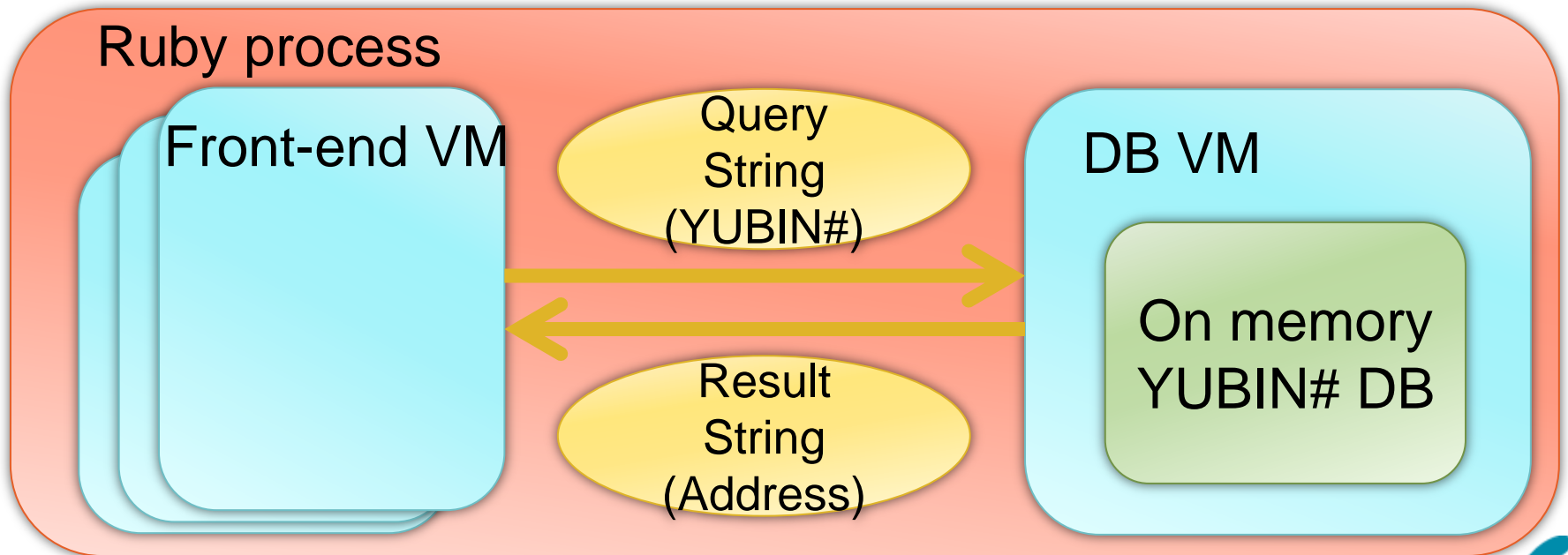
HTML rendering app



Evaluation

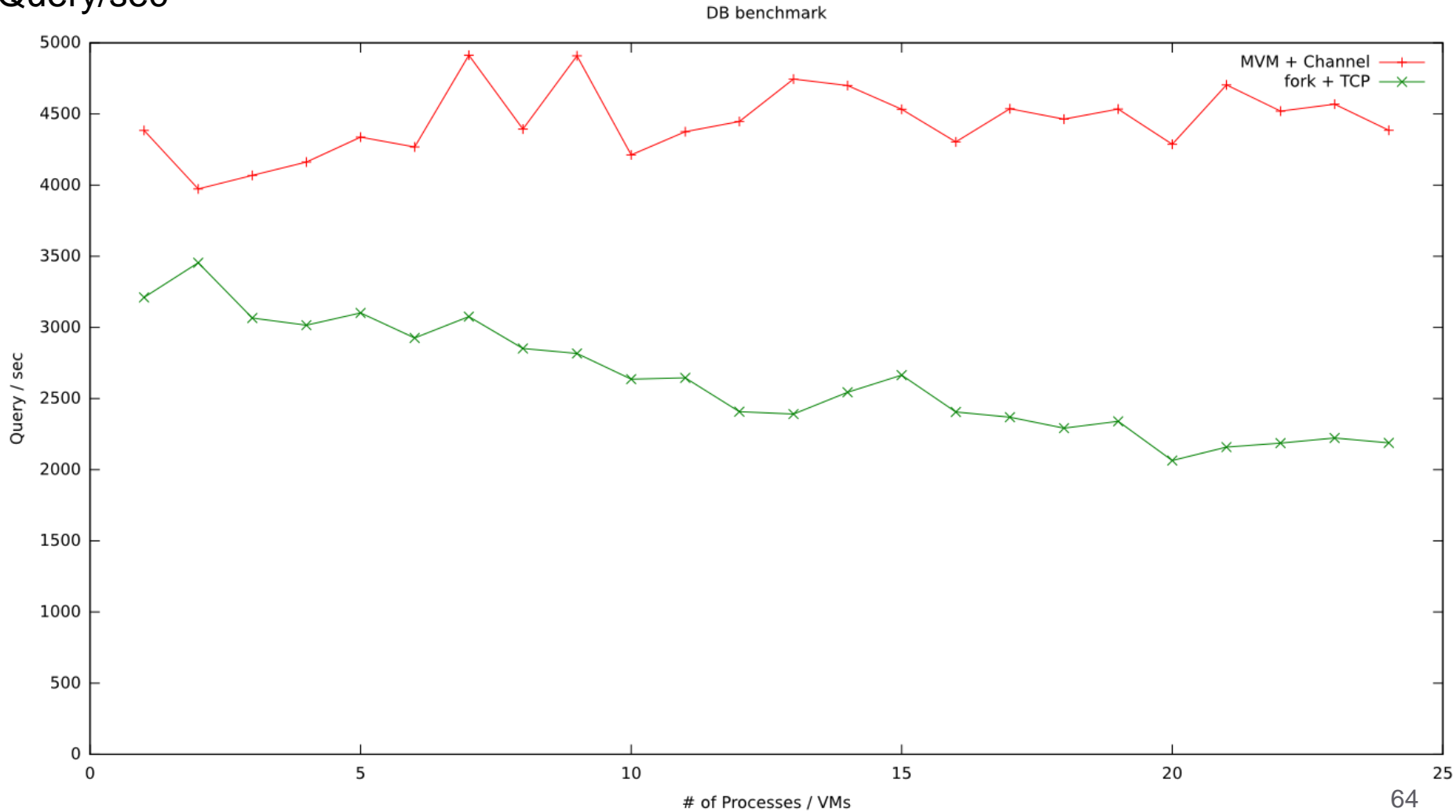
DB app

- Benchmark assuming web application
- Several front-end VMs and one DB VM
- YUBIN-Number (zip-code) DB on memory
- Using **dRuby** (w/MVM) framework



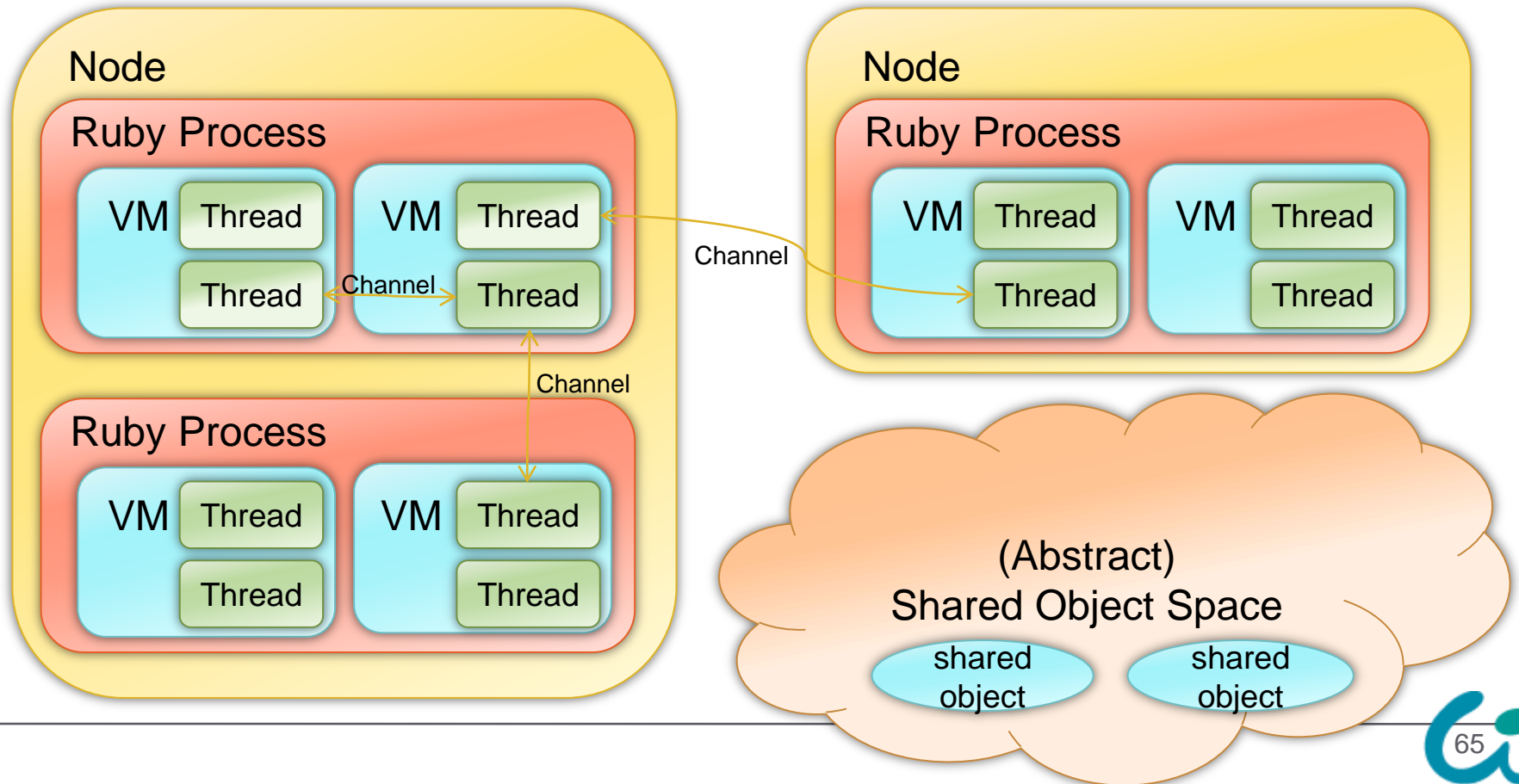
Evaluation DB app

Query/sec



Future work

- Extend this communication channel between inter-process (w/ shared memory), inter-node
- Migratable Ruby activity (threads, blocks (closures) and so on)



Summary

- ✓ Ruby 1.9.3 will be released soon!
 - ✓ Be Sacrifice!
 - ✓ NEWS file and blog posts are good to know changes
 - ✓ This talk introduces background on Ruby 1.9.3
- ✓ We are continuing the hacking to the future
 - ✓ Stay tuned to the next announcement

おしまい

Thank you for your attention

Implementation of Ruby 1.9.3 and later

ささだこういち

ko1@atdot.net, ko1@rvm.jp