mruby で mackerel の プラグインを作るはなし

今やるなら mruby

GMOペパボ株式会社

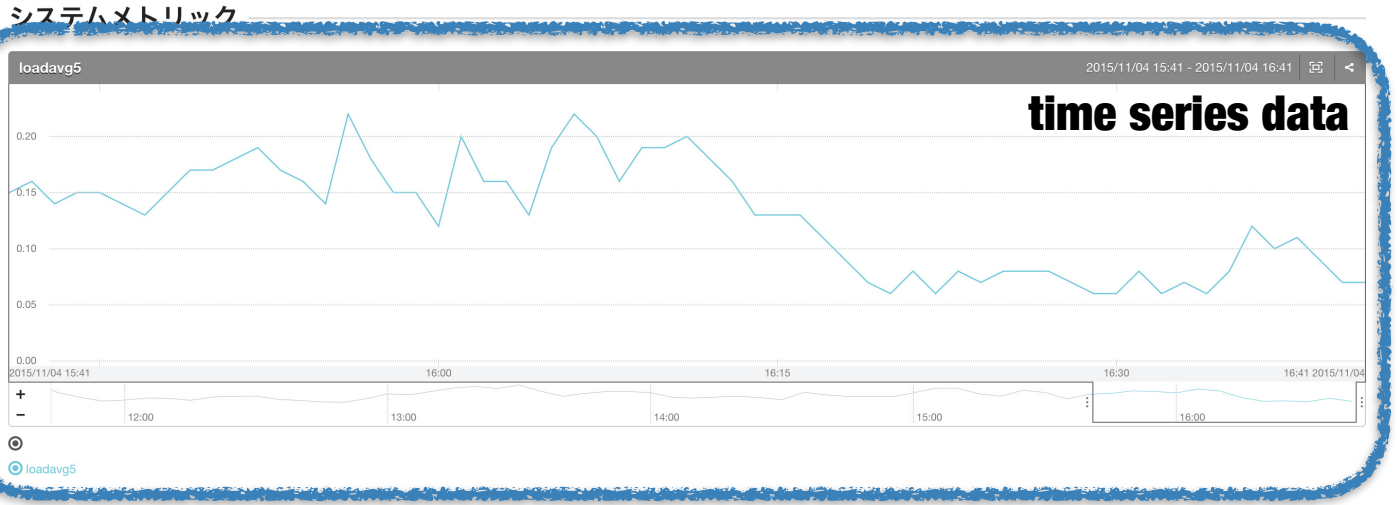


self.introduce

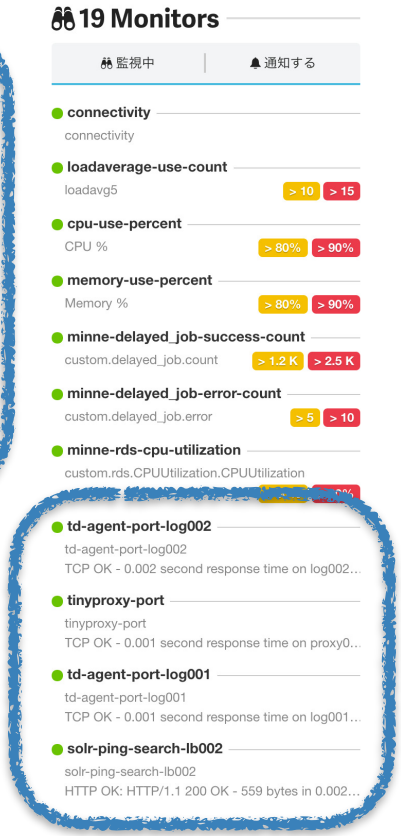


=>
{
  name: “SHIBATA Hiroshi”,
  nickname: “hsbt”,
  title: “Chief engineer at GMO Pepabo, Inc.”,
  commit_bits: [“ruby”, “rake”, “rubygems”, “rdoc”, “tdiary”, “hiki”, “railsgirls”, “railsgirls-jp”, …],
  sites: [“hsbt.org”, ruby-lang.org”, “rubyci.com”, “railsgirls.com”, “railsgirls.jp”],
}

What's mackerel plugin?







monitoring item

GMOペパボ株式会社

mackerel plugin format

- time series data
  - tab separated value at stdout

- monitoring item
  - nagios compatible results with exit status

Write cli tool using mruby-cli

see. https://github.com/hone/mruby-cli

1. Edit `build_config.rb` in top-level directory of mruby-cli
2. write mruby code into mrblib
3. build cli binary with docker or your development platform

I think it's too easily using mruby and mruby-redis at first.

NoMethodError…

```
> r = Redis.new '127.0.0.1', 6379
 => #<Redis:0x7fa109806230>
> r.scard
(mirb):3: undefined method 'scard' for #<Redis:0x7fa109806230> (NoMethodError)
> r.smember
(mirb):4: undefined method 'smember' for #<Redis:0x7fa109806230> (NoMethodError)
```

???

… mruby-redis is not support `Set` type function in redis.

We can make it

Reading mruby-redis

Checking out “https://github.com/matsumoto-r/mruby-redis” and open Rakefile

```
MRUBY_CONFIG=File.expand_path(ENV["MRUBY_CONFIG"] || "build_config.rb")
(snip)
file :mruby do
 (snip)
 sh "git clone --depth=1 git://github.com/mruby/mruby.git"
end

desc "compile binary"
task :compile => :mruby do
  sh "cd mruby && MRUBY_CONFIG=#{MRUBY_CONFIG} rake all"
end
(snip)
```

build_config.rb

```
MRuby::Build.new do |conf|
  # load specific toolchain settings
  toolchain :gcc

  (snip)

  conf.gembox 'default'
  conf.gem File.expand_path(File.dirname(__FILE__))
  conf.gem :github => 'matsumoto-r/mruby-sleep'

(snip)

end
```

Static link with hiredis

```
MRuby::Gem::Specification.new('mruby-redis') do |spec|
(snip)
  hiredis_dir = "#{build_dir}/hiredis"

  (snip)

  if ! File.exists? hiredis_dir
    Dir.chdir(build_dir) do
      e = {}
      run_command e, 'git clone git://github.com/redis/hiredis.git'
    end
  end
  (snip)
  spec.cc.include_paths << "#{hiredis_dir}/include"
  spec.linker.flags_before_libraries << “#{hiredis_dir}/lib/libhiredis.a"
  (snip)
end
```

Initialization point of native ext of mrbgem

see doc/guides/mrbgems.md on mruby/mruby

```
mrb_YOURGEMNAME_gem_init(mrb_state)

ex. ruby-redis: mrb_mruby_redis_gem_init(mrb_state)
```

In src/mruby_redis.h on matsumoto-r/mruby-redis

```
#ifndef MRB_REDIS_H
#define MRB_REDIS_H

void mrb_mruby_redis_gem_init(mrb_state *mrb);

#endif
```

Basic pattern of mruby class in C

```
void mrb_mruby_redis_gem_init(mrb_state *mrb)
{
    struct RClass *redis;

    redis = mrb_define_class(mrb, "Redis", mrb->object_class);

    mrb_define_class_under(mrb, redis, "ConnectionError", E_RUNTIME_ERROR);

    mrb_define_method(mrb, redis, "initialize", mrb_redis_connect, MRB_ARGS_ANY());
    mrb_define_method(mrb, redis, "select", mrb_redis_select, MRB_ARGS_REQ(1));
    (snip)
    mrb_define_method(mrb, redis, "flushdb", mrb_redis_flushdb, MRB_ARGS_NONE());
    (snip)

}
```

Write instance methods using C

```
mrb_value mrb_redis_llen(mrb_state *mrb, mrb_value self)
{
    mrb_value key;
    mrb_int integer;
    redisContext *rc = DATA_PTR(self);

    mrb_get_args(mrb, "o", &key);
    redisReply *rr = redisCommand(rc,"LLEN %s", mrb_str_to_cstr(mrb, key));
    integer = rr->integer;
    freeReplyObject(rr);

    return  mrb_fixnum_value(integer);
}
```

Implementation of “scard” command

“scard" received key of target Set. It’s same as “llen”

```
mrb_get_args(mrb, "o", &key);
```

```
mrb_define_method(mrb, redis, "scard", mrb_redis_scard, MRB_ARGS_REQ(1));
```

Replaced “LLEN” to “SCARD”

```
redisReply *rr = redisCommand(rc,"SCARD %s", mrb_str_to_cstr(mrb, key));
```

SCARD return integer value of Set length

```
integer = rr->integer;
```

```
return  mrb_fixnum_value(integer);
```

Implementation of “smember” command

```
mrb_value mrb_redis_smembers(mrb_state *mrb, mrb_value self)
{
    int i;
    mrb_value array, key;
    redisContext *rc = DATA_PTR(self);

    mrb_get_args(mrb, "o", &key);
    redisReply *rr = redisCommand(rc, "SMEMBERS %s", mrb_str_to_cstr(mrb, key));
    if (rr->type == REDIS_REPLY_ARRAY) {
        array = mrb_ary_new(mrb);
        for (i = 0; i < rr->elements; i++) {
            mrb_ary_push(mrb, array, mrb_str_new(mrb, rr->element[i]->str, rr->element[i]->len));
        }
    } else {
        freeReplyObject(rr);
        return mrb_nil_value();
    }

    freeReplyObject(rr);

    return array;
}
```

mackerel-plugin-sidekiq-job-count

https://github.com/hsbt/mackerel-plugin-sidekiq-job-count

```
def __main__(argv)
  if argv[1] == "version"
    puts "v#{SidekiqJobCount::VERSION}"
  else
    r = Redis.new argv[1], argv[2].to_i
    namespace = argv[3]

    key = ""
    key += "#{namespace}:" if namespace
    key += 'queues'

    queues = r.smembers key
    enqueued = queues.map{|queue| "#{key[0...-1]}:#{queue}" }.map{|k| r.llen k }.inject(0){|s, v| s + v}

    r.close

    puts "sidekiq_job_count.enqueued\t#{enqueued}\t#{Time.now.to_i}"
  end
end
```

It works!!1

```
[user@manage001 ~]$ /usr/local/bin/mackerel-plugin-sidekiq-job-count redis_host_name 6379 sidekiq
sidekiq_job_count.enqueued	0	1446624944
```

