High Performance tDiary

tDiary Anniversary Party

GMOペパボ株式会社



tDiary は遅い

stackprof

tDiary endpoint for stackprof

index.rb

```
(snip)
if encoding_error.empty?
	@cgi = cgi
else
	@cgi = CGI::new(accept_charset: 'shift_jis')
	@cgi.params = cgi.params
end

request = TDiary::Request.new( ENV, @cgi )
status, headers, body = TDiary::Dispatcher.index.dispatch_cgi( request, @cgi )

TDiary::Dispatcher.send_headers( status, headers )
::Rack::Handler::CGI.send_body(body)
(snip)
```



tDiary endpoint for stackprof

index.rb

```
(snip)
if encoding_error.empty?
	@cgi = cgi
else
	@cgi = CGI::new(accept_charset: 'shift_jis')
	@cgi.params = cgi.params
end

request = TDiary::Request.new( ENV, @cgi )
status = headers = body = nil
StackProf.run(mode: :cpu, out: “path/to/stackprof-cpu-#{Time.now.to_i}.dump”) do
status, headers, body = TDiary::Dispatcher.index.dispatch_cgi( request, @cgi )
end
TDiary::Dispatcher.send_headers( status, headers )
::Rack::Handler::CGI.send_body(body)
(snip)
```



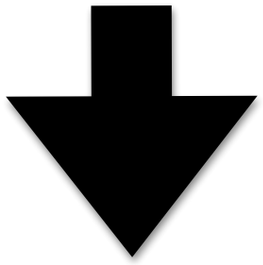
StackProf results

```
==================================
  Mode: cpu(1000)
  Samples: 36989 (6.05% miss rate)
  GC: 5689 (15.38%)
==================================
    TOTAL    (pct)     SAMPLES    (pct)     FRAME
     3293   (8.9%)        3290   (8.9%)     Dalli::Server#deserialize
     1497   (4.0%)        1497   (4.0%)     block (2 levels) in TDiary::IO::Default#calendar
     2733   (7.4%)        1371   (3.7%)     REXML::Attributes#get_attribute
     2549   (6.9%)        1145   (3.1%)      REXML::Parsers::BaseParser#pull_event
     2532   (6.8%)         931   (2.5%)     ERB::Compiler::SimpleScanner2#scan
     1861   (5.0%)         928   (2.5%)     REXML::Element#root
     1601   (4.3%)         865   (2.3%)     block in ERB::Compiler#compile
      838   (2.3%)         838   (2.3%)     block in REXML::Document#doctype
     8633  (23.3%)         619   (1.7%)      REXML::Element#namespace
      603   (1.6%)         603   (1.6%)     REXML::Source#match
      612   (1.7%)         594   (1.6%)     block in TDiary::Plugin#load_plugin
     5071  (13.7%)         579   (1.6%)      TDiary::IO::Default#transaction
      739   (2.0%)         572   (1.5%)     CGI::Util#unescape
      564   (1.5%)         535   (1.4%)     REXML::Text.check
(snip)
```



Dalli::Server#deserialize

memcached サーバーに cache を保存するやつ

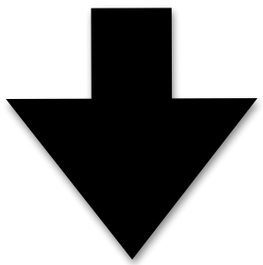


サーバー1台だと不要なので File + PStore に戻す



REXML::*

amazon.rb や flickr.rb の response をパース



native extension の Oga に書き換える



benchmark-ips

benchmark-ips

```
require 'benchmark/ips'
Benchmark.ips do |x|
  xml = File.read('../spec/fixtures/jpB00H91KK26.xml')
  require_relative '../misc/plugin/amazon'
  x.report('rexml') do
    item = AmazonItem.new(xml)
    amazon_detail_html( item )
  end
  x.report('oga') do
    require 'oga'
    item = AmazonItem.new(xml, :oga)
    amazon_detail_html(item)
  end
end
```



Results of Improvement

```
% ruby benchmark_amazon_plugin.rb
Warming up --------------------------------------
              rexml     2.000  i/100ms
               oga    14.000  i/100ms
Calculating -------------------------------------
              rexml     37.678  (±15.9%) i/s -    182.000  in   5.013022s
               oga    159.203  (±13.2%) i/s -    784.000  in   5.034065s
```

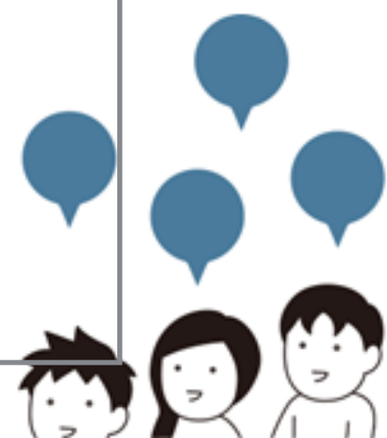
4.2 times faster!!1



Migration strategy for REXML and Oga

```
class AmazonItem
    def initialize(xml, parser = :rexml)
        @parser = parser
        if parser == :oga
            doc = Oga.parse_xml(xml)
            @item = doc.xpath('*/*/Item')[0]
        else
            doc = REXML::Document::new( REXML::Source::new( xml ) ).root
            @item = doc.elements.to_a( '*/Item' )[0]
        end
    end

    def nodes(path)
        if @parser == :oga
            @item.xpath(path)
        else
            @item.elements.to_a(path)
        end
    end
end
```



IO::Default#calendar

日記データをすべて参照して月のリストを作るメソッド、リクエストの度に毎回呼ばれている



tDiary のテストのしにくさよ...

ひたすら stub/mock module を用意する

```
class DummyTDiary
	attr_accessor :conf
	def initialize
		@conf = DummyConf.new
		@conf.data_path = TDiary.root + "/tmp/"
	end
	def ignore_parser_cache
		false
	end
end

class DummyConf
	attr_accessor :data_path
	def cache_path
		TDiary.root + "/tmp/cache"
	end
	def options
		{}
	end
	def style
		"wiki"
	end
end
```

```
module TDiary
	PATH = File::dirname( __FILE__ ).untaint
	class << self
		def root
			File.expand_path(File.join(library_root, '..'))
		end
		def library_root
			File.expand_path('..', __FILE__)
		end
		def server_root
			Dir.pwd.untaint
		end
	end
end
```



benchmark-ips(2)

```
conf = DummyConf.new
conf.data_path = TDiary.root + '/tmp/data/'
diary = DummyTDiary.new
diary.conf = conf
io = TDiary::IO::Default.new(diary)

require 'benchmark/ips'

Benchmark.ips do |x|
	x.report('calendar') do
		io.calendar
	end

	x.report('calendar2') do
		io.calendar2
	end
end
```



Improvement for calendar

```
def calendar
  calendar = {}
  Dir["#{@data_path}????"].sort.each do |dir|
    next unless %r[/\d{4}$] =~ dir
    Dir["#{dir.untaint}/??????.td2"].sort.each do |file|
      year, month = file.scan( %r[/(\d{4})(\d\d)\.td2$] )[0]
      next unless year
      calendar[year] = [] unless calendar[year]
      calendar[year] << month
    end
  end
  calendar
end
```

```
def calendar2
  calendar = {}
  Dir["#{@data_path}????/??????.td2"].sort.each do |file|
    if file =~ /(\d{4})(\d{2})\.td2$/
      calendar[$1] = [] unless calendar[$1]
      calendar[$1] << $2
    end
  end
  calendar
end
```



benchmark-ips

```
% ruby benchmark_io_default.rb
Warming up --------------------------------------
            calendar    16.000  i/100ms
           calendar2    22.000  i/100ms
Calculating -------------------------------------
            calendar    195.073  (±14.9%) i/s -    960.000  in   5.070197s
           calendar2    237.695  (±13.9%) i/s -      1.166k in   5.039136s
```

1.21 times faster!!1



Results and Conclusion

StackProf results(2)

```
==================================
  Mode: cpu(1000)
  Samples: 15391 (14.55% miss rate)
  GC: 2682 (17.43%)
==================================
    TOTAL    (pct)     SAMPLES    (pct)     FRAME
    43760 (284.3%)       3371  (21.9%)     TDiary::Plugin#load_plugin
     1014   (6.6%)       1014   (6.6%)     PStore#load
      487   (3.2%)        487   (3.2%)     <module:Flickr>
      445   (2.9%)        445   (2.9%)     TDiary::Configuration#[]
      768   (5.0%)        384   (2.5%)     TDiary::IO::Default#calendar
      374   (2.4%)        374   (2.4%)     CGI::Util#unescape
      614   (4.0%)        339   (2.2%)     TDiary::RefererManager#add_referer
      335   (2.2%)        335   (2.2%)     TDiary::Plugin#add_header_proc
      293   (1.9%)        293   (1.9%)     <module:Category>
     1798  (11.7%)         276   (1.8%)     Kernel#open
      224   (1.5%)        224   (1.5%)     TDiary::Plugin#enable_js
      220   (1.4%)        220   (1.4%)     TDiary::Plugin#add_conf_proc
     1620  (10.5%)         212   (1.4%)     TDiary::IO::Default#transaction
     1213   (7.9%)         199   (1.3%)     PStore#load_data
(snip)
```



速くなったかな…

CGI::Util.unescape

Ruby 2.4.0 で nobu が C ext で書き換えたので速くなる。2.2/2.3 向けに gem にバックポートしてもいいかも。



flickr プラグイン

amazon プラグイン同様に REXML が遅いので Oga に書き換える予定



TDiary::Plugin#load_plugin

プラグインを全部読み込んで eval してるアレ…

プラグイン機構を変えるとかして速くしたいなあ…

まずは eval の仕組みを利用すれば速くできそう

→プラグインを File.read する度に eval するのではなく全てを File.read して繋げてから eval すると速そう



速かった

```
Benchmark.ips do |x|
  o = Object.new
  x.report('multi-eval') do
    o.instance_eval("def a; puts 'a'; end")
(snip)
    o.instance_eval("def z; puts 'z'; end")
  end
  x.report('single-eval') do
    o.instance_eval("def a; puts 'a'; end; (snip) def z; puts 'z'; end")
  end
end
```

```
Warming up --------------------------------------
          multi-eval   283.000  i/100ms
         single-eval   565.000  i/100ms
Calculating -------------------------------------
          multi-eval      2.918k (±14.8%) i/s -     14.433k in   5.100649s
         single-eval      5.444k (±18.5%) i/s -     25.990k in   5.015969s
```



tDiary 楽しい!!1

15年やってもまだまだ hack しがいがある!!1

