Ruby - Feature #1628

GC.stat

06/14/2009 03:58 PM - ko1 (Koichi Sasada)

Status:	Closed
Priority:	Normal
Assignee:	ko1 (Koichi Sasada)
Target version:	2.0.0

Description

=begin
□ささだです.

□現在の GC 関連の状態を表示するためのメソッド GC.stat を追加するのはどうでしょうか.

{:count=>1, # これまで GC が起こった回数
:elapsed_time=>0.004, # これまで GC にかかったユーザ時間
:heap_used=>20, # 利用中の heap の数
:heap_length=>22, # 確保した heap の数
:heap_live_slot=>4742, # 生きている slot の数 (*1)
:heap_free_slot=>5056, # 空いている slot の数 (*1)
:heap_waiting_finalize_slot=>23} # ファイナライザを待っている slot の数

*1: 一番最後の GC でカウント.

Gauche では gc-stat というものがある，と shiro さんに聞いたもので.

gosh> (gc-stat)
((:total-heap-size 786432) (:free-bytes 163840) (:bytes-since-gc 532376)
(:total-bytes 532352))

Index: gc.c

--- gc.c (リビジョン 23689)
+++ gc.c (作業コピー)
@@ -323,6 +323,9 @@ typedef struct rb_objspace {
size_t length;
size_t used;
size_t longlife_used;

- size_t total_live_slot;
- size_t total_free_slot;
- size_t total_waiting_finalize_slot;
  RVALUE *freelist;
  RVALUE *longlife_freelist;
  RVALUE *range[2];
  @@ -355,7 +358,8 @@ typedef struct rb_objspace {
  double invoke_time;
  } profile;
  struct gc_list *global_list;

- unsigned int count;

- size_t count;
- double elapsed_time;
  int gc_stress;
  } rb_objspace_t;

@@ -933,7 +937,8 @@ assign_heap_slot(rb_objspace_t *objspace
if (lomem == 0 || lomem > p) lomem = p;
if (himem < pend) himem = pend;

if (lifetime == lifetime_longlife) objspace->heap.longlife_used++;

- heaps_used++;
- objspace->heap.used++;
- objspace->heap.total_free_slot += objs;

  while (p < pend) {
  p->as.free.flags = 0;
  @@ -1832,7 +1837,7 @@ static void
  gc_sweep(rb_objspace_t *objspace)
  {
  RVALUE *p, *pend, *final_list;

- size_t freed = 0;
- size_t freed = 0, waiting_finalize = 0;
  size_t i;
  size_t live = 0, free_min = 0, do_heap_free = 0;

@@ -1868,6 +1873,7 @@ gc_sweep(rb_objspace_t *objspace)
p->as.free.next = final_list;
final_list = p;
final_num++;

- [redacted]

```
}
else {
    add_freelist(objspace, &freelist, p);
```

@@ -1877,6 +1883,7 @@ gc_sweep(rb_objspace_t *objspace)
else if (BUILTIN_TYPE(p) == T_ZOMBIE) {
/* objects to be finalized */
/* do nothing remain marked */

- [redacted]

```
}
else {
RBASIC(p)->flags &= ~FL_MARK;
```

@@ -1923,6 +1930,10 @@ gc_sweep(rb_objspace_t *objspace)
free_unused_heaps(objspace);
GC_PROF_SET_HEAP_INFO;
}
+

- objspace->heap.total_live_slot = live;
- objspace->heap.total_free_slot = freed;
- objspace->heap.total_waiting_finalize_slot = waiting_finalize;
  }

static void
@@ -2207,6 +2218,8 @@ garbage_collect(rb_objspace_t *objspace)
{
struct gc_list *list;
rb_thread_t *th = GET_THREAD();

- double invoke_time = getrusage_time();
- INIT_GC_PROF_PARAMS;

  if (GC_NOTIFY) printf("start garbage_collect()\n");
  @@ -2286,6 +2299,8 @@ garbage_collect(rb_objspace_t *objspace)

GC_PROF_TIMER_STOP;
if (GC_NOTIFY) printf("end garbage_collect()\n");

- objspace->elapsed_time += getrusage_time() - invoke_time;

- return Qtrue;
}

@@ -2950,6 +2965,33 @@ gc_count(VALUE self)
return UINT2NUM((&rb_objspace)->count);
}

+/*

  - call-seq:

  - `GC.stat -> Hash`

  - Return information about GC.

  - It returns several information about GC:

  - total count, elapesed time, heap size, and so on.

- */
+static VALUE
+gc_stat(VALUE self)
+{
- VALUE hash = rb_hash_new();
- rb_objspace_t *objspace = &rb_objspace;

- rb_hash_aset(hash, ID2SYM(rb_intern("count")),
SIZET2NUM(objspace->count));
- rb_hash_aset(hash, ID2SYM(rb_intern("elapsed_time")),
DBL2NUM(objspace->elapsed_time));
- rb_hash_aset(hash, ID2SYM(rb_intern("heap_used")),
SIZET2NUM(objspace->heap.used));
- rb_hash_aset(hash, ID2SYM(rb_intern("heap_length")),
SIZET2NUM(objspace->heap.length));
- rb_hash_aset(hash, ID2SYM(rb_intern("heap_live_slot")),
SIZET2NUM(objspace->heap.total_live_slot));
- rb_hash_aset(hash, ID2SYM(rb_intern("heap_free_slot")),
SIZET2NUM(objspace->heap.total_free_slot));
- rb_hash_aset(hash, ID2SYM(rb_intern("heap_waiting_finalize_slot")),
SIZET2NUM(objspace->heap.total_waiting_finalize_slot));

- return hash;
+}

#if CALC_EXACT_MALLOC_SIZE
/*

- call-seq:
@@ -3122,6 +3164,7 @@ Init_GC(void)
rb_define_singleton_method(rb_mGC, "stress", gc_stress_get, 0);
rb_define_singleton_method(rb_mGC, "stress=", gc_stress_set, 1);
rb_define_singleton_method(rb_mGC, "count", gc_count, 0);

- rb_define_singleton_method(rb_mGC, "stat", gc_stat, 0);
rb_define_method(rb_mGC, "garbage_collect", rb_gc_start, 0);

rb_mProfiler = rb_define_module_under(rb_mGC, "Profiler");

--
// SASADA Koichi at atdot dot net
=end

History

#1 - 06/15/2009 06:23 PM - ko1 (Koichi Sasada)

=begin
□ささだです．

Tanaka Akira wrote::

> > {:count=>1, # これまで GC が起こった回数
> > :elapsed_time=>0.004, # これまで GC にかかったユーザ時間
> > :heap_used=>20, # 利用中の heap の数
> > :heap_length=>22, # 確保した heap の数
> > :heap_live_slot=>4742, # 生きている slot の数 (*1)
> > :heap_free_slot=>5056, # 空いている slot の数 (*1)
> > :heap_waiting_finalize_slot=>23} # ファイナライザを待っている slot の数
>
> heap という語は、Ruby 内部での意味とは異なる意味が一般に使われているので、わかりにくいのではないでしょうか。

□何がいいでしょう．例えば，slot_length, ... とかにして，live_slot とかにするとか．

□まぁ，heap* は，所詮 Ruby の実装を知ってる人しかわからないので，別にいいような気もするんですが．例えば，GC::Profiler なんかも，知らないとわからない内容な気がします．

--
// SASADA Koichi at atdot dot net

=end

#2 - 06/16/2009 12:52 AM - ko1 (Koichi Sasada)

=begin
□ささだです．

Tanaka Akira wrote::

> slot は、HEAP_MIN_SLOTS などをみると、オブジェクトの場所という意味で使われていて違うんじゃないかと思いますが。

□見てると，heap_slot というと，slot の集合のように使ってる感じですね．

□Lisp なんかだと，Cell なんて言いますけど，live cell とかは変な感じですよね．(live|free)_slot は，悪くないような気がしますが．

> > □まぁ，heap* は，所詮 Ruby の実装を知ってる人しかわからないので，別にいいような気もするんですが．例えば，GC::Profiler なんかも，知らないとわからない内容な気がします．
>
> 公開のメソッドにすれば、Ruby の実装を知らないひとが興味を持つことは充分に考えられるでしょう。
>
> 実装を知っているひとしかわからなくていいといういいわけは、より通用しにくくなるんじゃないでしょうか。

□個人的には，そんなに分かりづらい名前じゃないとは思います．よりよい名前の提案を頂ければ，それに越したことはないんですが．

□これを機会に，Ruby 実装での GC まわりの用語の整理とかをしてもらえると色々と便利かもしれません．

--
// SASADA Koichi at atdot dot net

=end

#3 - 06/16/2009 01:29 AM - matz (Yukihiro Matsumoto)

=begin
まつもと ゆきひろです

In message "Re: [ruby-dev:38642] Re: [Feature: trunk] GC.stat"
on Tue, 16 Jun 2009 00:52:03 +0900, SASADA Koichi ko1@atdot.net writes:

|▯見てると，heap_slot というと，slot の集合のように使ってる感じですね．

一応、名前を付けた本人は

heap - オブジェクトを切り出して来る空間
slot - 切り出してきたオブジェクト(の入れ物)

というニュアンスで名前付けてます。

=end

#4 - 06/16/2009 09:31 AM - ko1 (Koichi Sasada)

=begin
▯ささだです．

Yukihiro Matsumoto wrote::

> 一応、名前を付けた本人は
>
> heap - オブジェクトを切り出して来る空間
> slot - 切り出してきたオブジェクト(の入れ物)
>
> というニュアンスで名前付けてます。

▯slot の配列を heap，heap の配列を heaps ，でいいんでしたっけ．

▯そうすると，free_slot，live_slot（active_slot）は悪くないような．そも
そも heap が嫌って話?

--
// SASADA Koichi at atdot dot net

=end

#5 - 06/16/2009 09:59 AM - matz (Yukihiro Matsumoto)

=begin
まつもと ゆきひろです

In message "Re: [ruby-dev:38646] Re: [Feature: trunk] GC.stat"
on Tue, 16 Jun 2009 09:31:07 +0900, SASADA Koichi ko1@atdot.net writes:

|▯slot の配列を heap，heap の配列を heaps ，でいいんでしたっけ．

gc.cではそういう名前になってます。

|▯そうすると，free_slot，live_slot（active_slot）は悪くないような．そも
|そも heap が嫌って話?

一般的にはheapというのはメモリ割り当てで切り出してくる領域全
体を指すのではないかという指摘だと思います。Gaucheの例でも
(heapに)割り当てられたオブジェクトの数を返してますし。

=end

#6 - 06/16/2009 11:18 AM - ko1 (Koichi Sasada)

=begin
Yukihiro Matsumoto wrote::

> まつもと ゆきひろです
>
> In message "Re: [ruby-dev:38646] Re: [Feature: trunk] GC.stat"
> on Tue, 16 Jun 2009 09:31:07 +0900, SASADA Koichi ko1@atdot.net writes:
>
> |　slot の配列を heap，heap の配列を heaps ，でいいんでしたっけ．
>
> gc.cではそういう名前になってます。
>
> |　そうすると，free_slot，live_slot（active_slot）は悪くないような．そも
> |そも heap が嫌って話?
>
> 一般的にはheapというのはメモリ割り当てで切り出してくる領域全
> 体を指すのではないかという指摘だと思います。Gaucheの例でも
> (heapに)割り当てられたオブジェクトの数を返してますし。

　ちょっと考えたのですが，slot というのを，object_slot という形で明示す
れば，あまり間違いようがないと思うのですが，どうでしょう．

- active_object_slot: 生きてるオブジェクトが格納されている slot
- free_object_slot: 空き slot
- zombie_object_slot: ファイナライザ待ちオブジェクトの slot

　gc.c での heap の数（rb_objspace.heap.length）も，取りたいなぁ，と思い
始めたのですが，（heap を使用禁止だとすると）いい名前が思いつかない．

--
// SASADA Koichi at atdot dot net

=end

#7 - 06/16/2009 01:05 PM - matz (Yukihiro Matsumoto)

=begin
まつもと ゆきひろです

In message "Re: [ruby-dev:38652] Re: [Feature: trunk] GC.stat"
on Tue, 16 Jun 2009 11:18:38 +0900, SASADA Koichi ko1@atdot.net writes:

|　gc.c での heap の数（rb_objspace.heap.length）も，取りたいなぁ，と思い
|始めたのですが，（heap を使用禁止だとすると）いい名前が思いつかない．

heap(オブジェクトの割り当て空間)を構成するpageと考えれば名前
付けられるんじゃないでしょうか。

=end

#8 - 07/13/2009 09:55 PM - yugui (Yuki Sonoda)

- Assignee set to ko1 (Koichi Sasada)

- Target version set to 2.0.0

=begin

=end

#9 - 09/14/2010 04:26 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

=begin

=end

#10 - 10/15/2010 02:00 AM - ko1 (Koichi Sasada)

=begin
　ささだです。

(2009/06/14 7:57), SASADA Koichi wrote:

> 　現在の GC 関連の状態を表示するためのメソッド GC.stat を追加するのはど
> うでしょうか．

```
{:count=>1,                    # これまで GC が起こった回数
:elapsed_time=>0.004,         # これまで GC にかかったユーザ時間
:heap_used=>20,               # 利用中の heap の数
:heap_length=>22,             # 確保した heap の数
:heap_live_slot=>4742,      # 生きている slot の数 (*1)
:heap_free_slot=>5056,      # 空いている slot の数 (*1)
:heap_waiting_finalize_slot=>23} # ファイナライザを待っている slot の数
```

*1: 一番最後の GC でカウント.

Gauche では gc-stat というものがある，と shiro さんに聞いたもので.

```
gosh> (gc-stat)
((:total-heap-size 786432) (:free-bytes 163840) (:bytes-since-gc 532376)
(:total-bytes 532352))
```

　この件、キーの名前をどうするかでちゅうぶらりんになっていました。実装の詳細に踏み込んだ、処理系開発者しか見ないメソッドという意味を強調するために、GC.internal_statistic_information という感じで追加するのはどうでしょうか。

　で、構造体のフィールドの名前をキーとして追加する。

　rdoc には、

```
{:count=>1,                    # これまで GC が起こった回数
:elapsed_time=>0.004,         # これまで GC にかかったユーザ時間
```

しか載せず、もしかしたら他の情報も入っているかもしれないけれど、これはバージョン依存である、みたいな注釈があるような。

　どうでしょうか。

--
// SASADA Koichi at atdot dot net

=end

#11 - 10/15/2010 10:01 AM - matz (Yukihiro Matsumoto)

=begin
まつもと ゆきひろです

In message "Re: [ruby-dev:42385] Re: [Feature: trunk] GC.stat"
on Fri, 15 Oct 2010 02:00:43 +0900, SASADA Koichi ko1@atdot.net writes:

|　この件、キーの名前をどうするかでちゅうぶらりんになっていました。実装の
|詳細に踏み込んだ、処理系開発者しか見ないメソッドという意味を強調するため
|に、GC.internal_statistic_information という感じで追加するのはどうでしょ
|うか。

うーん、いくらなんでもそんな名前は嫌です。GC.statでいいじゃん。そこに反対したことはありません。

|　で、構造体のフィールドの名前をキーとして追加する。
|
|　rdoc には、
|
|> {:count=>1,   # これまで GC が起こった回数
|> :elapsed_time=>0.004,   # これまで GC にかかったユーザ時間
|
|しか載せず、もしかしたら他の情報も入っているかもしれないけれど、これは
|バージョン依存である、みたいな注釈があるような。

ドキュメントにGC.statは処理系依存で、戻り値は(最低限の共通項以外)バージョン依存であると明記するのでどうでしょうか。

=end

#12 - 10/15/2010 04:32 PM - ko1 (Koichi Sasada)

=begin

□ささだです。

(2010/10/15 2:00), Yukihiro Matsumoto wrote:

> |□この件、キーの名前をどうするかでちゅうぶらりんになっていました。実装の
> |詳細に踏み込んだ、処理系開発者しか見ないメソッドという意味を強調するため
> |に、GC.internal_statistic_information という感じで追加するのはどうでしょ
> |うか。
>
> うーん、いくらなんでもそんな名前は嫌です。GC.statでいいじゃ
> ん。そこに反対したことはありません。

□反対してたのは田中さんなんで...。

> ドキュメントにGC.statは処理系依存で、戻り値は(最低限の共通項
> 以外)バージョン依存であると明記するのでどうでしょうか。

□はい。それでいいなら、それが良いです。

--
// SASADA Koichi at atdot dot net

=end

#13 - 10/15/2010 04:58 PM - akr (Akira Tanaka)

=begin
2010年10月15日16:32 SASADA Koichi ko1@atdot.net:

>> |に、GC.internal_statistic_information という感じで追加するのはどうでしょ
>> |うか。
>>
>> うーん、いくらなんでもそんな名前は嫌です。GC.statでいいじゃ
>> ん。そこに反対したことはありません。
>
> □反対してたのは田中さんなんで...。

GC.stat に反対した覚えはありませんね。

反対なのは返り値の中に含まれる **heap** という語についてです。

[田中 哲][たなか あきら][Tanaka Akira]

=end

#14 - 10/15/2010 05:09 PM - ko1 (Koichi Sasada)

=begin
□ささだです。

(2010/10/15 8:58), Tanaka Akira wrote:

> 2010年10月15日16:32 SASADA Koichi ko1@atdot.net:
>
>>> |に、GC.internal_statistic_information という感じで追加するのはどうでしょ
>>> |うか。
>>>
>>> うーん、いくらなんでもそんな名前は嫌です。GC.statでいいじゃ
>>> ん。そこに反対したことはありません。
>>
>> □反対してたのは田中さんなんで...。
>
> GC.stat に反対した覚えはありませんね。
> 反対なのは返り値の中に含まれる heap という語についてです。

□はい。なので、

1. stat という使いやすい名前じゃなければいいのか、

2. ドキュメントにあればいいのか、
3. ソースを変えない限り駄目なのか、

って話で、1案を提案してみました。

□2案で良ければ、2案が良いです。

--
// SASADA Koichi at atdot dot net

=end

#15 - 10/18/2010 07:48 PM - akr (Akira Tanaka)

=begin
2010年10月15日17:09 SASADA Koichi ko1@atdot.net:

> > GC.stat に反対した覚えはありませんね。
> > 反対なのは返り値の中に含まれる heap という語についてです。
>
> □はい。なので、
>
> 1. stat という使いやすい名前じゃなければいいのか、

heap の問題は stat を変えても解決しませんねえ。

> 1. ドキュメントにあればいいのか、

注意書きはいいわけみたいなものでたいてい役に立たない、と思っています。

> 1. ソースを変えない限り駄目なのか、

どう変えるかによるんじゃないでしょうか。

[田中 哲][たなか あきら][Tanaka Akira]

=end

#16 - 10/18/2010 08:37 PM - ko1 (Koichi Sasada)

=begin
□ささだです。

(2010/10/18 11:48), Tanaka Akira wrote:

> > 1. stat という使いやすい名前じゃなければいいのか、
>
> heap の問題は stat を変えても解決しませんねえ。

□いいえ。

□田中さんの問題提起は、

> 公開のメソッドにすれば、Ruby の実装を知らないひとが興味を持
> つことは充分に考えられるでしょう。
>
> 実装を知っているひとしかわからなくていいといういいわけは、よ
> り通用しにくくなるんじゃないでしょうか。

ということで、「実装を知っている人しかわからなくていい」感を増やしたつも
りです。

> > 1. ドキュメントにあればいいのか、
>
> 注意書きはいいわけみたいなものでたいてい役に立たない、と思っています。

□という反論が来ていますが、まつもとさん、どうでしょうか。

□私は、言い訳でいいと思いますが。

--
// SASADA Koichi at atdot dot net

=end

#17 - 10/18/2010 08:57 PM - matz (Yukihiro Matsumoto)

=begin
まつもと ゆきひろです

In message "Re: [ruby-dev:42424] Re: [Feature: trunk] GC.stat"
on Mon, 18 Oct 2010 20:30:44 +0900, SASADA Koichi ko1@atdot.net writes:

|>> 2. ドキュメントにあればいいのか、
|>
|> 注意書きはいいわけみたいなものでたいてい役に立たない、と思っています。
|
|□という反論が来ていますが、まつもとさん、どうでしょうか。
|
|□私は、言い訳でいいと思いますが。

私も言い訳でいいと思います、少なくともこの件については。

=end

#18 - 10/28/2010 04:04 AM - ko1 (Koichi Sasada)

=begin
(2010/10/18 12:57), Yukihiro Matsumoto wrote:

> 私も言い訳でいいと思います、少なくともこの件については。

□というわけで、入れました。

- 言い訳を強調するためにエントリの名前を構造体の名前にあわせました。
- elapesed_time は、Profiler によりすぎなのでやめました。
（lazy sweep のことを考えると、ちょっと大変だし）

□こういう、細かいのも NEWS に書くんでしたっけ。

--
// SASADA Koichi at atdot dot net

=end

#19 - 10/28/2010 04:07 AM - ko1 (Koichi Sasada)

- Status changed from Assigned to Closed

- % Done changed from 0 to 100

=begin
This issue was solved with changeset r29616.
Koichi, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

=end