Ruby master - Bug #982

stack level too deep for long Array initialization

01/05/2009 10:56 AM - duerst (Martin Dürst)

Status:	Closed		
Priority:	Normal		
Assignee:	ko1 (Koichi Sasada)		
Target version:	2.0.0		
ruby -v:	-	**Backport:**	

Description

=begin
次のスクリプトが stack level too deep (SystemStackError) で終わります。

```
ruby -e 'puts "A=["; 0.upto(1000000) { puts "  [22, 55]," }; puts "]"' | ruby
```

これは Bug [#943](#943) [ruby-dev:37646] のコピーで、長期的にどうするのか
を考えるとめのバグです。

```
ruby -e 'puts "A=[]"; 0.upto(1000000) { puts "A<<[22, 55]" }' | ruby
```

で意図のものができるが、どうしてもデータの初期化を宣言的でできなく、
手続的にしないといけないのはよくないと思っています。ユーザから見ると
メゾドの深い呼び出しなどがスタックオーバフローになるのは分かるはず
ですが、データの初期化だけでそういう問題になるのはビックリする
でしょう。

笹田さんによりますと、

> []上記例のように，空の配列を作って，それに push するように変更することも
> 可能ですが，それを出来るようにしたほうがいいですかねぇ．するにしても，
> 1.9.2 で命令追加ってことになると思いますが．そう変更したら，ちょっと速度
> が遅くなるってくらいかなぁ．

もう一つのやり方は stack level too deep になったときに、スタックにある
配列の要素を一気に配列に追加し、次の要素からまだスタックに載せること
が考えられる。でも、こういうのを実装するのはどのぐらい難しいでしょうか。

よろしくお願いします。 Martin.
=end

Related issues:

Related to Ruby master - Bug #4040: SystemStackError with Hash[*a] for Large _a_	**Assigned**

History

#1 - 01/07/2009 02:20 AM - mame (Yusuke Endoh)

=begin
遠藤です。

> 次のスクリプトが stack level too deep (SystemStackError) で終わります。
>
> ruby -e 'puts "A=["; 0.upto(1000000) { puts " [22, 55]," }; puts "]"' | ruby

どういうときにこれが必要になったのか興味があります。

それはそれとして、パッチを書いてみました。

- concatarray 命令を廃止して、引数をとる concatarrays 命令を導入した
- checkints 命令を導入した
- 巨大な配列リテラルは 1000 個ごとに newarray して最後に concatarrays する

とりあえず配列だけの対処なので、巨大なハッシュリテラルを書くとたぶん落ちます。

Index: insns.def
===
--- insns.def (revision 21356)
+++ insns.def (working copy)
@@ -40,6 +40,20 @@
/* none */
}

+/**

- @c check interrupts
- @e check interrupts
- @j
- */ +DEFINE_INSN +checkints +() +() +() +{
- RUBY_VM_CHECK_INTS(); +} + /***/ / **deal with variables** */ / ***/ @@ -486,31 +500,27 @@

/**
@c put

- @e concat two arrays
- @j 二つの配列 ary1, ary2 を連結しスタックへプッシュする。
- @e put concatenate arrays
- @j スタックトップの配列を n 個連結し，結果をスタックにプッシュする。 */ DEFINE_INSN -concatarray -() -(VALUE ary1, VALUE ary2st) -(VALUE ary) +concatarrays +(rb_num_t num) +(...) +(VALUE val) // inc += 1 - num; {
- const VALUE ary2 = ary2st;
- VALUE tmp1 = rb_check_convert_type(ary1, T_ARRAY, "Array", "to_a");
- VALUE tmp2 = rb_check_convert_type(ary2, T_ARRAY, "Array", "to_a");

- int i;
- if (NIL_P(tmp1)) {
- tmp1 = rb_ary_new3(1, ary1);
- val = rb_ary_new();
- for (i = num - 1; i >= 0; i--) {
- const VALUE v = TOPN(i);
- VALUE tmp = rb_check_convert_type(v, T_ARRAY, "Array", "to_a");
- if (NIL_P(tmp)) {
- [redacted]
- }
- rb_ary_concat(val, tmp);

  }

- if (NIL_P(tmp2)) {
- tmp2 = rb_ary_new3(1, ary2);
- **}**
- if (tmp1 == ary1) {
- tmp1 = rb_ary_dup(ary1);
- }
- ary = rb_ary_concat(tmp1, tmp2);
- POPN(num);
  }

/**
Index: compile.c

```
==================================================================
--- compile.c  (revision 21356)
+++ compile.c  (working copy)
@@ -2195,12 +2195,14 @@
return COMPILE_OK;
}

+#define ARRAY_LITERAL_SPLIT_THRESHOLD 1000
+
static int
-compile_array_(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE* node_root,
```

- VALUE opt_p, int poped) +compile_array(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE* node_root, int poped) { NODE *node = node_root;
- int len = node->nd_alen, line = nd_line(node), i=0;
- int len = node->nd_alen, line = nd_line(node), i=0, split = 0;

- VALUE opt_p = Qtrue;
  DECL_ANCHOR(anchor);

  INIT_ANCHOR(anchor);
  @@ -2211,6 +2213,11 @@
  ruby_node_name(nd_type(node)));
  }
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
  i++;
  if (opt_p && nd_type(node->nd_head) != NODE_LIT) {
  opt_p = Qfalse;

  @@ -2243,17 +2250,36 @@
  }
  else {
  if (!poped) {
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

  }
  APPEND_LIST(ret, anchor);
  }
  return len;
  }

```
-static VALUE
-compile_array(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE* node_root, VALUE opt_p)
+static long
+compile_list(rb_iseq_t *iseq, LINK_ANCHOR *ret, NODE* node_root)
{
```

- return compile_array_(iseq, ret, node_root, opt_p, 0);
- NODE *node = node_root;
- int i = 0; +
- if (nd_type(node) != NODE_ZARRAY) {
- while (node) {
- if (nd_type(node) != NODE_ARRAY) {
- rb_bug("compile_list: This node is not NODE_ARRAY, but %s",

- ruby_node_name(nd_type(node)));
- }
- i++;
- COMPILE(ret, "list element", node->nd_head);
- node = node->nd_next;
- }
- } +
- return i; }

static VALUE
@@ -2841,8 +2867,7 @@
*flag |= VM_CALL_ARGS_SPLAT_BIT;

```
    if (next_is_array) {
```

- argc = INT2FIX(compile_array(iseq, args, argn->nd_head, Qfalse) + 1);
- POP_ELEMENT(args);
- argc = INT2FIX(compile_list(iseq, args, argn->nd_head) + 1); } else { argn = argn->nd_head; @@ -2851,8 +2876,7 @@ break; } case NODE_ARRAY: {
- argc = INT2FIX(compile_array(iseq, args, argn, Qfalse));
- POP_ELEMENT(args);

- [redacted]

```
break;

}
default: {
```

@@ -2862,10 +2886,7 @@
}

if (nsplat > 1) {

- int i;

- for (i=1; i<nsplat; i++) {

- [redacted]

- }

- ADD_INSN1(args_splat, nd_line(args), concatarrays, INT2FIX(nsplat));
}

if (!LIST_SIZE_ZERO(args_splat)) {
@@ -3746,7 +3767,7 @@
COMPILE(ret, "NODE_OP_ASGN1 args->head: ", node->nd_args->nd_head);
if (flag & VM_CALL_ARGS_SPLAT_BIT) {
ADD_INSN1(ret, nd_line(node), newarray, INT2FIX(1));

- [redacted]

- [redacted]

```
ADD_SEND_R(ret, nd_line(node), ID2SYM(idASET),
       argc, Qfalse, LONG2FIX(flag));
}
```

@@ -3769,7 +3790,7 @@
ADD_SEND(ret, nd_line(node), ID2SYM(id), INT2FIX(1));
if (flag & VM_CALL_ARGS_SPLAT_BIT) {
ADD_INSN1(ret, nd_line(node), newarray, INT2FIX(1));

- [redacted]

- [redacted]

```
ADD_SEND_R(ret, nd_line(node), ID2SYM(idASET),
       argc, Qfalse, LONG2FIX(flag));
}
```

@@ -4070,7 +4091,7 @@
ADD_INSN1(args, nd_line(node), getlocal, INT2FIX(idx));
}
ADD_INSN1(args, nd_line(node), newarray, INT2FIX(j));

- [redacted]

- [redacted]

```
    /* argc is setteled at above */
    }
    else {
```

@@ -4098,7 +4119,7 @@
break;
}
case NODE_ARRAY:{

- compile_array_(iseq, ret, node, Qtrue, poped);

- compile_array(iseq, ret, node, poped);
break;
}
case NODE_ZARRAY:{
@@ -4127,8 +4148,7 @@
INIT_ANCHOR(list);
switch (type) {
case NODE_ARRAY:{

- [redacted]

- [redacted]

- [redacted]

```
ADD_SEQ(ret, list);
break;
```

}
@@ -4426,14 +4446,14 @@
case NODE_ARGSCAT:{
COMPILE(ret, "argscat head", node->nd_head);
COMPILE(ret, "argscat body", node->nd_body);

- ADD_INSN(ret, nd_line(node), concatarray);

- ADD_INSN1(ret, nd_line(node), concatarrays, INT2FIX(2));
break;
}
case NODE_ARGSPUSH:{
COMPILE(ret, "arsgpush head", node->nd_head);
COMPILE(ret, "argspush body", node->nd_body);
ADD_INSN1(ret, nd_line(node), newarray, INT2FIX(1));

- ADD_INSN(ret, nd_line(node), concatarray);

- ADD_INSN1(ret, nd_line(node), concatarrays, INT2FIX(2));
break;
}
case NODE_SPLAT:{

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#2 - 04/27/2010 09:18 PM - mame (Yusuke Endoh)

- Target version changed from 1.9.2 to 2.0.0

- ruby -v set to -

=begin
遠藤です。

ささださんが「1.9.2 には不要ではないでしょうか」との見解を出したので、
target を 1.9.x とします。当面は Array#<< などを使ってください。

何とかして欲しいと言う声が多数あれば、早めに直されるかもしれません。

--
Yusuke Endoh mame@tsg.ne.jp
=end

#3 - 09/14/2010 04:44 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

=begin

=end

#4 - 11/12/2010 06:46 PM - duerst (Martin Dürst)

=begin

> どういうときにこれが必要になったのか興味があります。

具体的に言いますと、GB-18030 の変換テーブルのためです。
=end

#5 - 06/26/2011 05:31 PM - nahi (Hiroshi Nakamura)

- Target version changed from 2.0.0 to 1.9.3

#6 - 06/26/2011 07:29 PM - ko1 (Koichi Sasada)

- Target version changed from 1.9.3 to 2.0.0

すみません，1.9.3 の後の課題とさせて下さい．

#7 - 04/12/2012 10:51 AM - ko1 (Koichi Sasada)

- Status changed from Assigned to Closed

なおしました．
命令をいじるのではなく，core メソッドを呼ぶようにしてみました．
ちょっと，余計な配列が増えちゃうのが良くないかも….
誰か性能評価してくれると助かります．

#8 - 04/12/2012 07:00 PM - shugo (Shugo Maeda)

前田です。

ko1 (Koichi Sasada) wrote:

> なおしました．
> 命令をいじるのではなく，core メソッドを呼ぶようにしてみました．
> ちょっと，余計な配列が増えちゃうのが良くないかも….
> 誰か性能評価してくれると助かります．

内容はぜんぜん追っていないのですが、r35306のcommitから以下のような警告が出るようになっているようです。

```
compile.c: 関数 'iseq_compile_each' 内:
compile.c:4363:8: 警告: 変数 'size' が設定されましたが使用されていません [-Wunused-but-set-variable]
```

たしかに使われなくなっているように見えるのですが、これはこういうものでしょうか。