Ruby master - Feature #5839

Proposal: Bitmap Marking GC

01/04/2012 09:33 PM - authorNari (Narihiro Nakamura)

Status:	Closed	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:	2.0.0	

Description

あけましておめでとうございます。nariです。

ビットマップマーキングGCをRuby2.0向けに作りました。

ソースコード: https://github.com/authorNari/ruby/tree/bitmap_marking
パッチ: https://github.com/authorNari/patch_bag/blob/master/ruby/gc_bitmap_using_alignment_r33786.patch

以下の環境でr33786 に対するパッチで make check が通ること、
make TESTS="--gc-stress" test-all が無事に動いてることを確認してます。

$ ruby -v
ruby 2.0.0dev (2011-11-18 trunk 33786) [x86_64-linux]

= 性能評価

== make benchmark
make benchmark OPTS="-r 5" の結果は以下です。
https://gist.github.com/1542547
全体的に実行時間は若干遅くなるようです。

マークでRVALUEのフラグを立てるだけだったのが、ビットマップ上のフ
ラグを立てるようになるので遅くなるのはしょうがないかなと...。

== skkzipcode
ビットマップマーキングではマークでRVALUEに対して書き込みがなくなるので、
Linuxで複数の子プロセスを実行した場合にも無駄なCoWが発生せず、総メモリ
使用量が少なくなるという利点があります。

skkzipcodeというサンプルプログラムで上記の点を確認しました。
skkzipcodeは、親プロセスでたくさんメモリを使ってデータを保持しておいて、
生成した子プロセスで親プロセスと共有したデータを利用します。
https://github.com/authorNari/skkzipcode
（/proc/PID/smapsを使って計測しています）

origin - ビットマップマーキングなし
PROCESS_CNT : 5
SHARED_TOTAL: 59124 kb
PRIV_TOTAL : 224892 kb

bmap - ビットマップマーキングあり
PROCESS_CNT : 5
SHARED_TOTAL: 170744 kb
PRIV_TOTAL : 138336 kb

PROCESS_CNTは子プロセスの数、SHARED_TOTALは子プロセスで利用している共有
メモリ量、PRIV_TOTALは私有メモリ量です。bmapの方が共有メモリを沢山使っ
ていることがわかり、私有メモリ使用量も少なくなっています。

= 実装
実装について簡単にまとめます。

- ビットマップ探索高速化のため、ヒープ1ブロックのアドレスは16KBでアラインメントしている -- Linuxではposix_memalign(),memalign()を利用 -- Windowsでは_aligned_malloc()を利用
- 無駄な書き込みを防ぐため、なるべくfreelistを繋ぎ変えないようにした -- GCの時にすでにfreelistに繋ながれていたオブジェクトを再度繋ぎ直さない

- freelistはスロットが持つようになった -- 不要なスロットを解放するときにグローバルなfreelistにオブジェクトが繋がれている可能性がでてくるため
- struct heaps_slotをヒープブロックに埋め込んだ

Linuxでfork()を使わないようなプログラムに対してはビットマップマーキング
の恩恵はありません。ただ、CRubyで並列プログラミングしようとするときには
fork()を使って云々するライブラリがけっこう多いので（例えばpassengerとか…）
そういうものに対しては結構効くはずです。

GCの実行時間もそれほど遅くなりませんので、fork()を使わないプログラムで
もそれほど問題にならないと思っています。

Related issues:

Related to Ruby master - Bug #5901: OpenBSD "[FATAL] failed to allocate memory"	**Closed**	**01/17/2012**

Associated revisions

Revision 50675fdb - 01/07/2012 02:02 PM - nari

- gc.c: use Bitmap Marking algorithm to avoid copy-on-write of memory pages. See [ruby-dev:45085] [Feature #5839] [ruby-core:41916].
- include/ruby/ruby.h : FL_MARK rename to FL_RESERVED1.
- node.h : ditto.
- debug.c : ditto.
- object.c (rb_obj_clone): FL_MARK move to a bitmap.
- class.c (rb_singleton_class_clone): ditto.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@34225 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 34225 - 01/07/2012 02:02 PM - nari

- gc.c: use Bitmap Marking algorithm to avoid copy-on-write of memory pages. See [ruby-dev:45085] [Feature #5839] [ruby-core:41916].
- include/ruby/ruby.h : FL_MARK rename to FL_RESERVED1.
- node.h : ditto.
- debug.c : ditto.
- object.c (rb_obj_clone): FL_MARK move to a bitmap.
- class.c (rb_singleton_class_clone): ditto.

Revision 34225 - 01/07/2012 02:02 PM - nari

- gc.c: use Bitmap Marking algorithm to avoid copy-on-write of memory pages. See [ruby-dev:45085] [Feature #5839] [ruby-core:41916].
- include/ruby/ruby.h : FL_MARK rename to FL_RESERVED1.
- node.h : ditto.
- debug.c : ditto.

- object.c (rb_obj_clone): FL_MARK move to a bitmap.
- class.c (rb_singleton_class_clone): ditto.

Revision 34225 - 01/07/2012 02:02 PM - nari

- gc.c: use Bitmap Marking algorithm to avoid copy-on-write of memory pages. See [ruby-dev:45085] [Feature #5839] [ruby-core:41916].
- include/ruby/ruby.h : FL_MARK rename to FL_RESERVED1.
- node.h : ditto.
- debug.c : ditto.
- object.c (rb_obj_clone): FL_MARK move to a bitmap.
- class.c (rb_singleton_class_clone): ditto.

Revision 34225 - 01/07/2012 02:02 PM - nari

- gc.c: use Bitmap Marking algorithm to avoid copy-on-write of memory pages. See [ruby-dev:45085] [Feature #5839] [ruby-core:41916].
- include/ruby/ruby.h : FL_MARK rename to FL_RESERVED1.
- node.h : ditto.
- debug.c : ditto.
- object.c (rb_obj_clone): FL_MARK move to a bitmap.
- class.c (rb_singleton_class_clone): ditto.

Revision 34225 - 01/07/2012 02:02 PM - nari

- gc.c: use Bitmap Marking algorithm to avoid copy-on-write of memory pages. See [ruby-dev:45085] [Feature #5839] [ruby-core:41916].
- include/ruby/ruby.h : FL_MARK rename to FL_RESERVED1.
- node.h : ditto.
- debug.c : ditto.
- object.c (rb_obj_clone): FL_MARK move to a bitmap.
- class.c (rb_singleton_class_clone): ditto.

Revision 34225 - 01/07/2012 02:02 PM - nari

- gc.c: use Bitmap Marking algorithm to avoid copy-on-write of

memory pages. See [ruby-dev:45085] [Feature #5839]
[ruby-core:41916].

- include/ruby/ruby.h : FL_MARK rename to FL_RESERVED1.
- node.h : ditto.
- debug.c : ditto.
- object.c (rb_obj_clone): FL_MARK move to a bitmap.
- class.c (rb_singleton_class_clone): ditto.

History

#1 - 01/04/2012 10:53 PM - ko1 (Koichi Sasada)

□ささだです.

□bitmap marking 化ご苦労様です．2つ教えて下さい.

(2012/01/04 21:33), Narihiro Nakamura wrote:

> - 無駄な書き込みを防ぐため、なるべくfreelistを繋ぎ変えないようにした -- GCの時にすでにfreelistに繋ながれていたオブジェクトを再度繋ぎ直さない
> - freelistはスロットが持つようになった -- 不要なスロットを解放するときにグローバルなfreelistにオブジェクトが繋 がれている可能性がでてくるため

□freelist を見ずに，bitmap を見る，みたいなのだと，free obj 探索が遅く
なっちゃう感じでしょうか.

□あと，nari さんが PRO で提案していた手法だと，「ビットマップ探索高速化
のため〜」云々はあまり気にしなくていいんじゃないかと思ったんですが，そう
でもないでしょうか.

あと，REE との性能比較があると興味深いと思ったけど，
いろいろ難しいかな.

□パッチをちらっと見ただけでの将来的な要望ですが，この辺は gc.c の中で綺
麗に API（というかマクロ）で切り離せるところだと思うので，うまいこと整理
できるといいんでないかと思います（以前，RubyConf2011 でも喋った話ですが）.

--
// SASADA Koichi at atdot dot net

#2 - 01/04/2012 10:53 PM - matz (Yukihiro Matsumoto)

まつもと ゆきひろです

In message "Re: [ruby-dev:45086] Re: [ruby-trunk - Feature #5839][Open] Proposal: Bitmap Marking GC"
on Wed, 4 Jan 2012 22:33:03 +0900, SASADA Koichi ko1@atdot.net writes:

|(2012/01/04 21:33), Narihiro Nakamura wrote:
|> - 無駄な書き込みを防ぐため、なるべくfreelistを繋ぎ変えないようにした
|> -- GCの時にすでにfreelistに繋ながれていたオブジェクトを再度繋ぎ直さない
|> - freelistはスロットが持つようになった
|> -- 不要なスロットを解放するときにグローバルなfreelistにオブジェクトが繋
|> がれている可能性がでてくるため
|
|□freelist を見ずに，bitmap を見る，みたいなのだと，free obj 探索が遅く
|なっちゃう感じでしょうか.

「bitmapあるからfreelistなくす」のもありえると思いますが、性
能特性はどうなるのかな。明示的なsweepが要らないぶんと割り当て
時にスキャンが発生するのとのトレードオフかなあ。

|□あと，nari さんが PRO で提案していた手法だと，「ビットマップ探索高速化
|のため〜」云々はあまり気にしなくていいんじゃないかと思ったんですが，そう
|でもないでしょうか.

memalignで直接ページが得られた方が、PROの手法よりビットマップ
テーブルを得るためのメモリアクセスが1段減って高速のはずです。
ビットマップテーブル取得はオブジェクトごとに発生しますから効
いてくるはずです。

|# あと，REE との性能比較があると興味深いと思ったけど，
|# いろいろ難しいかな．

うーん、1.8と1.9の差の方が大きすぎて意味のある性能比較はムリ
でしょう。メモリ消費量比較くらいなら有意義な比較ができるかな。

まつもと ゆきひろ　/:|)

#3 - 01/05/2012 08:29 AM - ko1 (Koichi Sasada)

□ささだです．

(2012/01/04 22:48), Yukihiro Matsumoto wrote:

> 「bitmapあるからfreelistなくす」のもありえると思いますが、性
> 能特性はどうなるのかな。明示的なsweepが要らないぶんと割り当て
> 時にスキャンが発生するのとのトレードオフかなあ。

□メモリアクセスが減る（かもしれない）ので，実測値を知りたいところです．
まぁ，そのへんは実装によると思いますが．

> |□あと，nari さんが PRO で提案していた手法だと，「ビットマップ探索高速化
> |のため～」云々はあまり気にしなくていいんじゃないかと思ったんですが，そう
> |でもないでしょうか．
>
> memalignで直接ページが得られた方が、PROの手法よりビットマップ
> テーブルを得るためのメモリアクセスが１段減って高速のはずです。
> ビットマップテーブル取得はオブジェクトごとに発生しますから効
> いてくるはずです。

□そうですね．定量的な比較がもしあると説得力が増すと思いました．

> |# あと，REE との性能比較があると興味深いと思ったけど，
> |# いろいろ難しいかな．
>
> うーん、1.8と1.9の差の方が大きすぎて意味のある性能比較はムリ
> でしょう。メモリ消費量比較くらいなら有意義な比較ができるかな。

□はい，難しいと思います．が，何らかの指標があると，マーケティングには良
さそうです．

--
// SASADA Koichi at atdot dot net

#4 - 01/05/2012 09:53 AM - authorNari (Narihiro Nakamura)

nariです。

2012年1月5日8:26 SASADA Koichi ko1@atdot.net:

> □ささだです．
>
> (2012/01/04 22:48), Yukihiro Matsumoto wrote:
>
> > 「bitmapあるからfreelistなくす」のもありえると思いますが、性
> > 能特性はどうなるのかな。明示的なsweepが要らないぶんと割り当て
> > 時にスキャンが発生するのとのトレードオフかなあ。
>
> □メモリアクセスが減る（かもしれない）ので，実測値を知りたいところです．
> まぁ，そのへんは実装によると思いますが．

その辺りはどうなんでしょう…。実装するのはそれほど難しくないので、試し
に実装して計測してみます。

実装的には、sweep時に解放対象のオブジェクトに対してはobj_free()を呼ぶだ
け、flagsは0にしない、freelistも繋がない、スロットをsweepしたあとに
bitmapをクリアしない、オブジェクトアロケーション時にbitmapスキャン、み
たいなのを考えています。

> |□あと，nari さんが PRO で提案していた手法だと，「ビットマップ探索高速化

|[illegible]
|[illegible]

memalign[illegible]PRO[illegible]
[illegible]
[illegible]
[illegible]

[illegible]

[illegible]
[illegible]...[illegible]

|# [illegible]REE [illegible]
|# [illegible]

[illegible]1.8[illegible]1.9[illegible]
[illegible]

[illegible]
[illegible]

REE[illegible]
passenger[illegible]skkzipcode[illegible]

REE[illegible]GC[illegible]
[illegible]...[illegible]

[illegible] gc.c [illegible]
[illegible] API[illegible]
[illegible]RubyConf2011 [illegible]

[illegible]GC[illegible]
[illegible]
[illegible]
[illegible]

--
// SASADA Koichi at atdot dot net

--
Narihiro Nakamura (nari)

#5 - 01/05/2012 09:53 AM - ko1 (Koichi Sasada)

[illegible]

(2012/01/05 9:30), Narihiro Nakamura wrote:

[illegible]...[illegible]
[illegible]

[illegible]sweep[illegible]obj_free()[illegible]
[illegible]flags[illegible]0[illegible]freelist[illegible]sweep[illegible]
bitmap[illegible]bitmap[illegible]
[illegible]

[illegible]allocation [illegible]
[illegible]bitmap [illegible]
[illegible]

|[illegible]nari [illegible] PRO [illegible]
|[illegible]
|[illegible]

memalign[illegible]PRO[illegible]
[illegible]
[illegible]

いてくるはずです。

> そうですね．定量的な比較がもしあると説得力が増すと思いました．

実装して比較してもよいですが、実装者の感覚としては調べるまでもな
いような気も…。

> なるほど．自明＆大変そうなら結構です．

> パッチをちらっと見ただけでの将来的な要望ですが，この辺は gc.c の中で綺麗に API（というかマクロ）で切り離せるところだと思うので，うまいこと整理できるといいんでないかと思います（以前，RubyConf2011 でも喋った話ですが）．

そうですね。それはいろんなGCを選択したいなぁ、という時にやらないといけ
ないことだと思っています。
また、ささださんがおっしゃってる「うまいこと整理」のイメージがあまり湧
いてないので、どういうものか例を上げてくれると想像しやすいです。

> 例えば，mark 部分をマクロ化するとか，slot の allocation をマクロ化するとか，そんな感じです．ただ，もちろんいろんな切り口のある話だと思いますので色々悩みどころになるんじゃないかと思います．
>
> # ちなみに，GC を選択，というのは広範囲過ぎて危険な言葉ではないかと
> # 思っています．copy gc にするのは無理だろうし．
>
> --
> // SASADA Koichi at atdot dot net

#6 - 01/05/2012 12:53 PM - authorNari (Narihiro Nakamura)

nari です。

2012年1月5日9:48 SASADA Koichi ko1@atdot.net:

> ささだです．
>
> (2012/01/05 9:30), Narihiro Nakamura wrote:
>
> > その辺りはどうなんでしょう…。実装するのはそれほど難しくないので、試しに実装して計測してみます。
> >
> > 実装的には、sweep時に解放対象のオブジェクトに対してはobj_free()を呼ぶだけ、flagsは0にしない、freelistも繋がない、スロットをsweepしたあとにbitmapをクリアしない、オブジェクトアロケーション時にbitmapスキャン、みたいなのを考えています。
>
> > 個人的には，allocation のコストがどう変わるか，が気にあっています．bitmap を総スキャンすると大変なので，最後のカーソルを保存する，とかかなぁと思っていました．まぁ，導入された後で私がやってもいいですが．

では、おまかせします :)

> > |# あと，REE との性能比較があると興味深いと思ったけど，
> > |# いろいろ難しいかな．
> >
> > うーん、1.8と1.9の差の方が大きすぎて意味のある性能比較はムリでしょう。メモリ消費量比較くらいなら有意義な比較ができるかな。
>
> > はい，難しいと思います．が，何らかの指標があると，マーケティングには良さそうです．

REEとメモリ消費量を比較したのがあったほうがいいかもしれませんね。
passengerは面倒そうなので、まずはskkzipcodeを動かしてみます。

REEとRuby2.0ビットマーキングのメモリ使用量を比較しました。
https://gist.github.com/1542547#file_skkzipcode_mem_usage.txt

Ruby2.0はREEよりも省メモリなので、総メモリ使用量が変動していますが、
共有メモリ使用量と私有メモリ使用量の割合はほとんど変わりないようです。

> [] パッチをちらっと見ただけでの将来的な要望ですが，この辺は gc.c の中で綺麗に API（というかマクロ）で切り離せるところだと思うので，うまいこと整理できるといいんでないかと思います（以前，RubyConf2011 でも喋った話ですが）．

そうですね。それはいろんなGCを選択したいなぁ、という時にやらないといけないことだと思っています。
また、ささださんがおっしゃってる「うまいこと整理」のイメージがあまり湧いてないので、どういうものか例を上げてくれると想像しやすいです。

> []例えば，mark 部分をマクロ化するとか，slot の allocation をマクロ化するとか，そんな感じです．ただ，もちろんいろんな切り口のある話だと思いますので色々悩みどころになるんじゃないかと思います．

> ちなみに， **GC** を選択，というのは広範囲過ぎて危険な言葉ではないかと
>
> 思っています． **copy gc** にするのは無理だろうし．

理解力不足で申し訳ないのですが、何か具体的な目的はあるんでしょうか?
アロケーションの戦略を変えたり、マーキングの方法を変えたり、を簡単にしたいとかそういう話でしょうか? 例えば並列マーキングに簡単に切り替えられるとか、tcmallocが簡単に使えるようになるとか…。
個人的には具体的な目的があったほうがAPIも定義しやすいと思っています。

--
Narihiro Nakamura (nari)

#7 - 01/05/2012 12:58 PM - mattetti (Matt Aimonetti)

If Google translation doesn't fail me totally, the patch improves the memory usage of forked Ruby processed on Linux but the GC performance is affected in other cases making this patch not applicable at the moment.
Options to speed up the GC were discussed as well as ways to properly benchmark the effect for making the GC CoW friendly.

Is that correct?

Thanks,

- Matt

#8 - 01/05/2012 02:53 PM - matz (Yukihiro Matsumoto)

まつもと ゆきひろです

礼儀としてruby-coreで報告したら、マージしてもいいんじゃない?

In message "Re: [ruby-dev:45085] [ruby-trunk - Feature #5839][Open] Proposal: Bitmap Marking GC"
on Wed, 4 Jan 2012 21:33:17 +0900, Narihiro Nakamura authorNari@gmail.com writes:

|Issue #5839 has been reported by Narihiro Nakamura.
|
|--
|Feature #5839: Proposal: Bitmap Marking GC
|https://bugs.ruby-lang.org/issues/5839
|
|Author: Narihiro Nakamura
|Status: Open
|Priority: Normal
|Assignee: Yukihiro Matsumoto
|Category: core
|Target version: 2.0.0
|
|
|あけましておめでとうございます。nariです。
|
|ビットマップマーキングGCをRuby2.0向けに作りました。
|

|ソースコード: https://github.com/authorNari/ruby/tree/bitmap_marking
|パッチ: https://github.com/authorNari/patch_bag/blob/master/ruby/gc_bitmap_using_alignment_r33786.patch
|
|以下の環境でr33786 に対するパッチで make check が通ること、
|make TESTS="--gc-stress" test-all が無事に動いてることを確認してます。
|
|$ ruby -v
|ruby 2.0.0dev (2011-11-18 trunk 33786) [x86_64-linux]
|
|= 性能評価
|
|== make benchmark
|make benchmark OPTS="-r 5" の結果は以下です。
|https://gist.github.com/1542547
|全体的に実行時間は若干遅くなるようです。
|
|マークでRVALUEのフラグを立てるだけだったのが、ビットマップ上のフ
|ラグを立てるようになるので遅くなるのはしょうがないかなと…。
|
|== skkzipcode
|ビットマップマーキングではマークでRVALUEに対して書き込みがなくなるので、
|Linuxで複数の子プロセスを実行した場合にも無駄なCoWが発生せず、総メモリ
|使用量が少なくなるという利点があります。
|
|skkzipcodeというサンプルプログラムで上記の点を確認しました。
|skkzipcodeは、親プロセスでたくさんメモリを使ってデータを保持しておいて、
|生成した子プロセスで親プロセスと共有したデータを利用します。
|https://github.com/authorNari/skkzipcode
|（/proc/PID/smapsを使って計測しています）
|
|origin - ビットマップマーキングなし
|PROCESS_CNT : 5
|SHARED_TOTAL: 59124   kb
|PRIV_TOTAL : 224892   kb
|
|bmap - ビットマップマーキングあり
|PROCESS_CNT : 5
|SHARED_TOTAL: 170744   kb
|PRIV_TOTAL : 138336   kb
|
|PROCESS_CNTは子プロセスの数、SHARED_TOTALは子プロセスで利用している共有
|メモリ量、PRIV_TOTALは私有メモリ量です。bmapの方が共有メモリを沢山使っ
|ていることがわかり、私有メモリ使用量も少なくなっています。
|
|= 実装
|実装について簡単にまとめます。
|
|- ビットマップ探索高速化のため、ヒープ1ブロックのアドレスは16KBでアライ
| ンメントしている
|-- Linuxではposix_memalign(),memalign()を利用
|-- Windowsでは_aligned_malloc()を利用
|- 無駄な書き込みを防ぐため、なるべくfreelistを繋ぎ変えないようにした
|-- GCの時にすでにfreelistに繋ながれていたオブジェクトを再度繋ぎ直さない
|- freelistはスロットが持つようになった
|-- 不要なスロットを解放するときにグローバルなfreelistにオブジェクトが繋
| がれている可能性がでてくるため
|- struct heaps_slotをヒープブロックに埋め込んだ
|
|Linuxでfork()を使わないようなプログラムに対してはビットマップマーキング
|の恩恵はありません。ただ、CRubyで並列プログラミングしようとするときには
|fork()を使って云々するライブラリがけっこう多いので（例えばpassengerとか…）
|そういうものに対しては結構効くはずです。
|
|GCの実行時間もそれほど遅くなりませんので、fork()を使わないプログラムで
|もそれほど問題にならないと思っています。
|
|
|
|--
|http://redmine.ruby-lang.org

#9 - 01/06/2012 10:53 AM - ko1 (Koichi Sasada)

(2012/01/05 12:30), Narihiro Nakamura wrote:

> [illegible]
> [illegible]
> [illegible]
> [illegible]tcmalloc[illegible]…[illegible]
> [illegible]API[illegible]

[illegible]
[illegible]
[illegible]

[illegible]
[illegible]

--
// SASADA Koichi at atdot dot net

#10 - 01/06/2012 12:53 PM - authorNari (Narihiro Nakamura)

2012[illegible]1[illegible]6[illegible]10:39 SASADA Koichi :

> (2012/01/05 12:30), Narihiro Nakamura wrote:
>
> > [illegible]
> > [illegible]
> > [illegible]
> > [illegible]tcmalloc[illegible]…[illegible]
> > [illegible]API[illegible]
>
> [illegible]
> [illegible]
> [illegible]

[illegible]

> [illegible]
> [illegible]

[illegible] :P

--
Narihiro Nakamura (nari)

#11 - 01/07/2012 11:02 PM - authorNari (Narihiro Nakamura)

- Status changed from Open to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r34225.
Narihiro, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- gc.c: use Bitmap Marking algorithm to avoid copy-on-write of memory pages. See [ruby-dev:45085] [Feature #5839] [ruby-core:41916].
- include/ruby/ruby.h : FL_MARK rename to FL_RESERVED1.
- node.h : ditto.
- debug.c : ditto.
- object.c (rb_obj_clone): FL_MARK move to a bitmap.
- class.c (rb_singleton_class_clone): ditto.