It has been shown that increasing the malloc_limit can have a dramatic effect in speeding up certain programs. 

ex:

Ruby 1.9 normal:
real-world/bm_hilbert_matrix.rb,
4.0054624080658,4.09904861450195,4.052255511283875,0.046793103218079,50

ruby 1.9 trunk with increased malloc_limit [1]
real-world/bm_hilbert_matrix.rb,
2.87246918678284,2.8920271396637,2.882248163223267,0.009778976440430,50

So I would like to request getters and setters for malloc_limit, HEAP_MIN_SLOTS, etc, if that's possible. Thanks much!

--r

Martin Duerst wrote:

I agree that something in this direction should be done. Garbage collection details can affect performance a lot.

Exposing one or two variables isn’t difficult. I’m wondering why this hasn’t been done yet. One guess is that it would not at all be portable to other systems (JRuby,…).

No reason MRI should not consider making their GC tunable IMHO.

I was only trying to guess reasons why this hasn’t happened yet, and that was the only one I was able to come up with (but not a very good one, I agree).

Regards, Martin.

I think GC tunables at startup or during runtime should be considered implementation dependent. Java has many different GCs. Certainly not all GC’s have the same tunable parameters. Even if they did would you really expect a tunable to work on different GC impls as expected?

If this is doable at runtime, then it would be nice if the GC API would allow a name to be associated with the parameter being tweaked so we could ignore the setting if using the wrong GC. The name is just a random idea, but everyone should get the point...

If it is settable at the command-line...ditto...

Tom

I’d expand this to point out that all Java’s GCs have many, many possible settings as well, but almost all are configured at startup. There are very few settings that can be tweaked at runtime, for a couple reasons:

- Hotspot makes some decisions about how to allocate and reallocate heap space immediately at startup.
- Hotspot also adjust GC settings (heap ratios, tenuring rates, etc) based on runtime information. So tweaking things at runtime would be forcing profiled settings to be thrown out.

But in general, having some set of command-line configurable settings
would be a great idea, if it could be done without a performance impact.
JRuby users are becoming more accustomed to having many available config
settings when they need them.

- Charlie

#4 - 02/02/2009 12:38 PM - ko1 (Koichi Sasada)
- Assignee set to authorNari (Narihiro Nakamura)

#5 - 03/05/2010 05:10 AM - adgar (Michael Edgar)
- File 1047.patch added

Hi,

I've attached a patch adding getters and setters for HEAP_MIN_SLOTS and GC_MALLOC_LIMIT. All tests still pass, and the ruby-benchmark-suite shows no slowdown.

It works by simply adding two extra static variables that is initially defined to the currently-existing constants, and using that instead of the compile-time constants throughout the code. I then add getters and setters retrieve and modify those variables.

#6 - 03/18/2010 09:16 PM - mame (Yusuke Endoh)
- Status changed from Open to Feedback
- Target version set to 2.0.0

Hi,

I've attached a patch adding getters and setters for HEAP_MIN_SLOTS and GC_MALLOC_LIMIT.

My personal opinions:

- Though I think a few rough "modes" is acceptable (e.g., -server and -clients of Java), such a specific parameter tuning is not Ruby way.

- The name `malloc_limit' is very arguable because it is too specific to implementation-detail of MRI.

- HEAP_MIN_SLOTS is just referred when the interpreter is initialized. So I guess dynamic change of HEAP_MIN_SLOTS makes no sense.

- I cannot reproduce performance improvement as shown by Roger. Could you show me a self-contained code to run benchmark?

Thanks,

--
Yusuke Endoh mame@tsg.ne.jp

#7 - 03/18/2010 09:54 PM - murphy (Kornelius Kalnbach)

On 18.03.10 13:16, Yusuke Endoh wrote:

My personal opinions:

- Though I think a few rough "modes" is acceptable (e.g., -server and -clients of Java), such a specific parameter tuning is not Ruby way. But it may be the only viable way to provide GC options independent of the Ruby implementation. HEAP_MIN_SLOTS is also an implementation detail (I think).

Another way would be to mimic --server and --client on MRI in a way that
optimizes the GC as expected (--client optimizes for startup speed and code running once, --server optimizes for long-running and background processes).

[murphy]
=end

#8 - 03/19/2010 01:46 PM - headius (Charles Nutter)
=begin
On Thu, Mar 18, 2010 at 7:16 AM, Yusuke Endoh redmine@ruby-lang.org wrote:

- The name `malloc_limit' is very arguable because it is too specific to implementation-detail of MRI.

I really hope we would not start to see a bunch of libraries or frameworks setting things like "malloc_limit". You probably worry the same thing.

- HEAP_MIN_SLOTS is just referred when the interpreter is initialized. So I guess dynamic change of HEAP_MIN_SLOTS makes no sense.

FWIW, on the JVM, most of the GC tunable settings must be set on startup as well.

- Charlie
=end

#9 - 03/20/2010 01:45 AM - rogerdpack (Roger Pack)
=begin
- File bm_hilbert_matrix.rb added
- Status changed from Feedback to Assigned

Attaaching demo file.
A little contrived still, but on my box, the things that help it improve are (believe it or not)

-#define HEAP_MIN_SLOTS 10000
+#define HEAP_MIN_SLOTS 100000

12.3s normal gc.c

to 10.4s

Perhaps defining more space up front makes it not use N^2 garbage traversals as it grows? It appears that's useful when you know you're going to be needing the space eventually anyway...in reality you could also put it as the default and it wouldn't hurt "small scripts" too much either, I'd imagine.

changing MALLOC_LIMIT has been shown effective in rails apps though I don't have any immediate numbers handy [1]

As a side note, with 1.9.1 (default) it takes 14.8s, so some improvement already--thanks guys!

If we do eventually go to a --server --client model, --server could include some other optimizations, too, like lookup cacheing

http://redmine.ruby-lang.org/issues/show/641

While we're on the subject of GC optimization, you might be some speedup by making gc_stress settings only available if GC_DEBUG is defined--I highly doubt gc_stress is used much in the wild, though I could be proven wrong.

Cheers!
-rp

=end

#10 - 03/20/2010 02:22 AM - mame (Yusuke Endoh)
=begin
Hi,

2010/3/20 Roger Pack redmine@ruby-lang.org:

A little contrived still, but on my box, the things that help it improve are (believe it or not)
Thanks. 24.2s -> 16.5s on my environment.

I actually tried with ruby-benchmark-suite and Michael's patch. But I couldn't confirm improvement by setting GC.malloc_limit. Now I'm sure that in this case, the performance improvement seems to be achieved by changing HEAP_MIN_SLOTS, rather than MALLOC_LIMIT.

If we do eventually go to a --server --client model, --server could include some other optimizations, too, like lookup cacheing

Note that the model is just my opinion :-)
Hi,

Well then, how do you think we can accomplish this performance improving without depending too much on MRI internals.

I myself think configuration via environment variables or command line options can rely on internals.

matz.

In message "Re: [ruby-core:35440] [Ruby 1.9 - Feature #1047] request: getters, setters for the GC" on Mon, 7 Mar 2011 11:11:13 +0900, Narihiro Nakamura authorNari@gmail.com writes:

| Hi. |
| I think this patch depend too much on inside implementation of MRI. |
| So, I can't agree this patch... |
| |
| Thanks. |

Hi,

2011/3/7 Yukihiro Matsumoto matz@ruby-lang.org:

Hi,

Well then, how do you think we can accomplish this performance improving without depending too much on MRI internals.

I myself think configuration via environment variables or command line options can rely on internals.

OK. I intend to think about specifications in reference to this patch.

Thanks.

--
Narihiro Nakamura (nari)

I think the desired goal is a "user configurable" GC, whatever that may mean finally (I'm ok with direct internals).

Cheers!

-f

Nari, any update on this?

--
Yusuke Endoh mame@tsg.ne.jp

We have introduced the GC parameter configuration to trunk at r31044.

Thanks.
<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Last Modified</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>1047.patch</td>
<td>3.51 KB</td>
<td>03/05/2010</td>
<td>adgar (Michael Edgar)</td>
</tr>
<tr>
<td>bm_hilbert_matrix.rb</td>
<td>682 Bytes</td>
<td>03/20/2010</td>
<td>rogerdpack (Roger Pack)</td>
</tr>
<tr>
<td>patch-1.9.2-gc.patch</td>
<td>3.32 KB</td>
<td>03/06/2011</td>
<td>funny_falcon (Yura Sokolov)</td>
</tr>
<tr>
<td>patch-1.9.2-gc-methods.patch</td>
<td>3.21 KB</td>
<td>03/06/2011</td>
<td>funny_falcon (Yura Sokolov)</td>
</tr>
<tr>
<td>patch-1.9.2-gc-methods.patch</td>
<td>3.21 KB</td>
<td>03/06/2011</td>
<td>funny_falcon (Yura Sokolov)</td>
</tr>
</tbody>
</table>