Ruby master - Misc #16124

Let the transient heap belong to objspace

08/24/2019 12:41 PM - methodmissing (Lourens Naudé)

Status:	Assigned	
Priority:	Normal	
Assignee:	ko1 (Koichi Sasada)	

Description

As per comment from Nobu in https://github.com/ruby/ruby/pull/2303#issuecomment-523248875 , I took an initial stab @ a tighter integration between objspace and the transient heap in https://github.com/ruby/ruby/pull/2400

Benefits

- Multi-VM (MVM) friendly - (vm -> objspace -> theap)
- The 32MB (current size) arena lazy allocated on ruby init is now properly freed on shutdown as well
- It feels strange that the evacuation from the current global theap is to objspace, whereas the space evacuated from is a global arena.

Not so great

- A fast reference to a global variable global_transient_heap becomes a function call to rb_objspace_get_theap() and related pointer chasing from vm -> objspace -> theap
- Some internal transient heap structs moved to the header file now leaks into all other reference sites where this source file (transient_heap.c) as previously just used for API
- I'm not sure exactly of the boundary Koichi had in mind for the GC compile module and how tightly it should (or shouldn't) be coupled to the transient heap. struct rb_objspace* declarations elsewhere for example reveals nothing about the structure members for example, whereas with this PR a lot of transient heap internals are exposed via the header file now
- Also possible to move transient_heap.c into gc.c - I feel theap is not an experimental feature anymore and has been stable for quite some time with plausible performance benefits. The downside of that is gc.c is quite dense already, but then all ruby heap management concerns belong to one compile unit.

In a similar vein the global method cache could perhaps belong to the VM instance as well, effectively better alignment with MVM and also easier to have a balanced VM setup and teardown sequence without anything left dangling on ruby shutdown.

Thoughts?

History

#1 - 08/24/2019 12:43 PM - methodmissing (Lourens Naudé)

The global method cache note references PR https://github.com/ruby/ruby/pull/2302

#2 - 08/24/2019 03:04 PM - mame (Yusuke Endoh)

- Assignee set to ko1 (Koichi Sasada)

- Status changed from Open to Assigned

#3 - 09/19/2019 07:52 AM - nobu (Nobuyoshi Nakada)

I'm positive about this, except for the performance.
Do you have any numbers?

#4 - 09/22/2019 02:36 PM - methodmissing (Lourens Naudé)

nobu (Nobuyoshi Nakada) wrote:

> I'm positive about this, except for the performance.
> Do you have any numbers?

Using the rdoc gc bench tooling Koichi used in https://bugs.ruby-lang.org/issues/14858, it's difficult to get consistent results between this change and master.

Master:

```
lourens@CarbonX1:~/src/ruby/trunk$ make gcbench-rdoc
```

```
Script: ./benchmark/gc/rdoc.rb
{:count=>137,
 :heap_allocated_pages=>10124,
 :heap_sorted_length=>10124,
 :heap_allocatable_pages=>0,
 :heap_available_slots=>4126494,
 :heap_live_slots=>4117009,
 :heap_free_slots=>9485,
 :heap_final_slots=>0,
 :heap_marked_slots=>2396930,
 :heap_eden_pages=>10124,
 :heap_tomb_pages=>0,
 :total_allocated_pages=>10124,
 :total_freed_pages=>0,
 :total_allocated_objects=>24630985,
 :total_freed_objects=>20513976,
 :malloc_increase_bytes=>23046392,
 :malloc_increase_bytes_limit=>32225676,
 :minor_gc_count=>113,
 :object_id_collisions=>0,
 :major_gc_count=>24,
 :remembered_wb_unprotected_objects=>2482,
 :remembered_wb_unprotected_objects_limit=>4964,
 :old_objects=>2387069,
 :old_objects_limit=>4774138,
 :oldmalloc_increase_bytes=>48011656,
 :oldmalloc_increase_bytes_limit=>41096563}
ruby 2.7.0dev (2019-09-22T07:39:47Z master 2272efa463) [x86_64-linux] ["USE_RGENGC", "RGENGC_DEBUG", "RGENGC_E
STIMATE_OLDMALLOC", "GC_ENABLE_LAZY_SWEEP"]

./benchmark/gc/rdoc.rb
      user     system      total        real
 18.842867   0.687732  19.530599 ( 19.722540)
GC total time (sec): 0

VmHWM: 411940 kB

Summary of rdoc on 2.7.0dev 19.722539804002736  0   137
         (real time in sec, GC time in sec, GC count)
```

Objspace linked theap:

```
lourens@CarbonX1:~/src/ruby/ruby$ make gcbench-rdoc
Script: ./benchmark/gc/rdoc.rb
{:count=>137,
 :heap_allocated_pages=>10122,
 :heap_sorted_length=>10122,
 :heap_allocatable_pages=>0,
 :heap_available_slots=>4125809,
 :heap_live_slots=>4116170,
 :heap_free_slots=>9639,
 :heap_final_slots=>0,
 :heap_marked_slots=>2396677,
 :heap_eden_pages=>10122,
 :heap_tomb_pages=>0,
 :total_allocated_pages=>10122,
 :total_freed_pages=>0,
 :total_allocated_objects=>24630955,
 :total_freed_objects=>20514785,
 :malloc_increase_bytes=>23055352,
 :malloc_increase_bytes_limit=>32225676,
 :minor_gc_count=>113,
 :object_id_collisions=>0,
 :major_gc_count=>24,
 :remembered_wb_unprotected_objects=>2482,
 :remembered_wb_unprotected_objects_limit=>4964,
 :old_objects=>2387070,
 :old_objects_limit=>4774140,
 :oldmalloc_increase_bytes=>47981528,
 :oldmalloc_increase_bytes_limit=>41096563}
ruby 2.7.0dev (2019-09-22T11:43:20Z objspace-theap 11b7839f09) [x86_64-linux] ["USE_RGENGC", "RGENGC_DEBUG", "
RGENGC_ESTIMATE_OLDMALLOC", "GC_ENABLE_LAZY_SWEEP"]

./benchmark/gc/rdoc.rb
      user     system      total        real
```

```
 18.517571   0.764057  19.281628 ( 19.303723)
GC total time (sec): 0

VmHWM: 411780 kB

Summary of rdoc on 2.7.0dev 19.303723024990177  0   137
         (real time in sec, GC time in sec, GC count)
```

Array specific (adapted from the original issue to include RUBY_DESCRIPTION):

```
p RUBY_DESCRIPTION
require 'benchmark'
N = 10_000_000

def n_times str, args = ''
  eval <<-EOS
    proc{|max, #{args}|
      i = 0
      while i < max
        #{str}
        i+=1
      end
    }
  EOS
end

m = n_times 'ary = Array.new(size)', 'size'

Benchmark.bm(10){|x|
  0.step(to: 16){|i|
    size = i
    x.report(size){
      m.call(N, size)
    }
  }
}
```

objspace-theap branch:

```
lourens@CarbonX1:~/src/ruby/ruby$ ./miniruby -Ilib $HOME/src/theap_array.rb
"ruby 2.7.0dev (2019-09-22T11:43:20Z objspace-theap 11b7839f09) [x86_64-linux]"
                 user     system      total        real
0            1.060482   0.000000   1.060482 (  1.060477)
1            1.063371   0.000000   1.063371 (  1.063398)
2            1.049839   0.000000   1.049839 (  1.049846)
3            1.066307   0.000000   1.066307 (  1.066311)
4            1.187135   0.000000   1.187135 (  1.187147)
5            1.130953   0.000000   1.130953 (  1.130964)
6            1.160799   0.000000   1.160799 (  1.160805)
7            1.220265   0.000000   1.220265 (  1.220272)
8            1.160544   0.000000   1.160544 (  1.160554)
9            1.262271   0.000000   1.262271 (  1.262282)
10           1.182926   0.000000   1.182926 (  1.182933)
11           1.305176   0.000000   1.305176 (  1.305187)
12           1.170598   0.000000   1.170598 (  1.170605)
13           1.243933   0.000000   1.243933 (  1.243940)
14           1.197929   0.000000   1.197929 (  1.197955)
15           1.257225   0.000000   1.257225 (  1.257232)
16           1.203359   0.000000   1.203359 (  1.203365)
```

master:

```
lourens@CarbonX1:~/src/ruby/trunk$ ./miniruby -Ilib $HOME/src/theap_array.rb
"ruby 2.7.0dev (2019-09-22T07:39:47Z master 2272efa463) [x86_64-linux]"
                 user     system      total        real
0            1.034709   0.000000   1.034709 (  1.035330)
1            1.073345   0.000000   1.073345 (  1.073350)
2            1.046693   0.000000   1.046693 (  1.046704)
3            1.046481   0.000000   1.046481 (  1.046487)
4            1.135405   0.000000   1.135405 (  1.135410)
5            1.152307   0.000000   1.152307 (  1.152308)
6            1.163179   0.000000   1.163179 (  1.163184)
7            1.229176   0.000000   1.229176 (  1.229193)
8            1.171530   0.000000   1.171530 (  1.171541)
9            1.229018   0.000000   1.229018 (  1.229023)
```

```
10           1.218360   0.000000   1.218360 (  1.218369)
11           1.258088   0.000000   1.258088 (  1.258093)
12           1.214547   0.000000   1.214547 (  1.214557)
13           1.246675   0.000000   1.246675 (  1.246693)
14           1.217621   0.000000   1.217621 (  1.217620)
15           1.240967   0.000000   1.240967 (  1.240972)
16           1.246610   0.000000   1.246610 (  1.246620)
```

Optcarrot (noisy too):

```
lourens@CarbonX1:~/src/optcarrot$ benchmark-driver -e "objspace-theap::~/src/ruby/ruby/ruby -I~/src/ruby/ruby/
lib -I~/src/ruby/ruby/. -I~/src/ruby/ruby/.ext/x86_64-linux" -e "trunk::~/src/ruby/trunk/ruby -I~/src/ruby/tru
nk/lib -I~/src/ruby/trunk/. -I~/src/ruby/trunk/.ext/x86_64-linux" --repeat-count 24 --output=all -v benchmark.
yml
objspace-theap: ruby 2.7.0dev (2019-09-22T11:43:20Z objspace-theap 11b7839f09) [x86_64-linux]
trunk: ruby 2.7.0dev (2019-09-22T07:39:47Z master 2272efa463) [x86_64-linux]
Calculating -------------------------------------
                          objspace-theap                 trunk
           optcarrot    42.40538956057937     42.50095691198821 fps
                        42.98025714388297     43.19539965591333
                        43.17850049383824     43.69380960879121
                        44.34221997472645     43.75925724088885
                        45.23015773087887     43.95609333352798
                        45.60259428038163     44.30873259117270
                        45.62534786154934     44.50561054139629
                        45.69848233271295     44.74074528123102
                        45.72577876475985     44.75584411276473
                        45.82588019886050     44.90017657928921
                        45.84436354482752     44.93827167130445
                        46.72210774150135     45.32808901780013
                        46.78934236282241     45.40725425997227
                        46.83007726756269     45.50752517888226
                        46.84959507374096     45.51210655013940
                        46.93537135184541     45.51427328345052
                        47.20234803912213     45.61616028100521
                        47.36550159135397     45.90162191663695
                        47.95694822288893     46.18908487592959
                        48.00296030568151     46.19919623309980
                        49.21328383054638     46.47376631857036
                        49.23558095502081     46.85201420357048
                        49.30196173221466     47.39761548726396
                        50.07004688586536     47.70501205501753
```

#5 - 11/18/2019 08:48 AM - ko1 (Koichi Sasada)

> Some internal transient heap structs moved to the header file now leaks into all other reference sites where this source file (transient_heap.c) as previously just used for API

not a big issue, but I don't want to expose them...