Ruby - Bug #20695

Elevated GC allocations in parse.y parser

08/23/2024 10:03 PM - alanwu (Alan Wu)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.4.0dev (2024-08-23T14:49:27Z master 3f6be01bfc) [arm64-darwin23]	**Backport:**	3.1: DONTNEED, 3.2: DONTNEED, 3.3: DONTNEED

Description

Testing on the lobsters benchmark from yjit-bench, the latest master parser is allocating more objects compared to 98eeadc932 ("Development of 3.4.0 started."). The following patch shows allocation count after code loading:

```
diff --git a/benchmarks/lobsters/benchmark.rb b/benchmarks/lobsters/benchmark.rb
index 240c50c..6cdd0ac 100644
--- a/benchmarks/lobsters/benchmark.rb
+++ b/benchmarks/lobsters/benchmark.rb
@@ -7,6 +7,8 @@ Dir.chdir __dir__
 use_gemfile

 require_relative 'config/environment'
+printf "allocated_after_load=%d\n", GC.stat(:total_allocated_objects)
+exit
 require_relative "route_generator"

 # For an in-mem DB, we need to load all data on every boot

$ ruby benchmarks/lobsters/benchmark.rb
ruby 3.4.0dev (2023-12-25T09:13:40Z master 98eeadc932) [arm64-darwin23]
<snip>
allocated_after_load=1747084
$ chruby master
$ ruby benchmarks/lobsters/benchmark.rb
ruby 3.4.0dev (2024-08-23T14:49:27Z master 3f6be01bfc) [arm64-darwin23]
<snip>
allocated_after_load=2163031
$ ruby --parser=prism benchmarks/lobsters/benchmark.rb
ruby 3.4.0dev (2024-08-23T14:49:27Z master 3f6be01bfc) +PRISM [arm64-darwin23]
<snip>
allocated_after_load=932571
$ math 2163031 / 1747084
1.238081
```

Profiling shows extra GC allocations coming from set_number_literal(); set_yylval_node() expands to a call to rb_enc_str_new(). So this issue seems related to #20659. A surprising side effect of these extra allocations is that they seem to impact the speed of the benchmark, not only the loading speed, even though the body of benchmark *does not do any Ruby parsing* after warmup iterations. The speed impact seem to be due to poorer data locality, as perf(1) shows elevated levels of CYCLE_ACTIVITY.STALLS_L2_MISS and related events. Using Prism speeds up benchmark iterations, probably since it allocates a lot fewer during code loading, leaving the heap in a better layout in terms of locality. With Prism, fewer cycles stall on data cache misses.

```
$ ruby run_benchmarks.rb --chruby 'master;mstr+p::master --parser=prism' lobsters
<snip>
master: ruby 3.4.0dev (2024-08-23T14:49:27Z master 3f6be01bfc) [arm64-darwin23]
mstr+p: ruby 3.4.0dev (2024-08-23T14:49:27Z master 3f6be01bfc) +PRISM [arm64-darwin23]

--------  -----------  ----------  -----------  ----------  --------------  -------------
bench     master (ms)  stddev (%)  mstr+p (ms)  stddev (%)  mstr+p 1st itr  master/mstr+p
lobsters  567.9        0.7         557.0        0.8         1.00            1.02
--------  -----------  ----------  -----------  ----------  --------------  -------------
Legend:
```

```
- mstr+p 1st itr: ratio of master/mstr+p time for the first benchmarking iteration.
- master/mstr+p: ratio of master/mstr+p time. Higher is better for mstr+p. Above 1 represents a sp
eedup.
```

In hindsight, it's clear how that the parser's allocation pattern can have a significant impact on heap layout, since typical app boot with Kernel#require alternates between running app code and running the parser. Parser allocations essentially end up as gaps between app objects.

Associated revisions

Revision c93d07ed7448f332379cf21b4b7b649b057e5671 - 09/02/2024 11:40 PM - yui-knk (Kaneko Yuichiro)

[Bug #20695] Do not create needless string object in parser

set_parser_s_value does nothing in parser therefore no need to
create string object in parser set_yylval_node.

Object allocation

Run ruby benchmarks/lobsters/benchmark.rb with the patch

```
diff --git a/benchmarks/lobsters/benchmark.rb b/benchmarks/lobsters/benchmark.rb
index 240c50c..6cdd0ac 100644
--- a/benchmarks/lobsters/benchmark.rb
+++ b/benchmarks/lobsters/benchmark.rb
@@ -7,6 +7,8 @@ Dir.chdir __dir__
 use_gemfile

 require_relative 'config/environment'
+printf "allocated_after_load=%d\n", GC.stat(:total_allocated_objects)
+exit
 require_relative "route_generator"

 # For an in-mem DB, we need to load all data on every boot
```

Before

```
ruby 3.4.0dev (2024-08-31T18:30:25Z master d6fc8f3d57) [arm64-darwin21]
...
allocated_after_load=2143519
```

After

```
ruby 3.4.0dev (2024-09-01T00:40:04Z fix_bugs_20695 d1bae52f75) [arm64-darwin21]
...
allocated_after_load=1579662
```

Ruby 3.3.0 for reference

```
ruby 3.3.0 (2023-12-25 revision 5124f9ac75) [arm64-darwin21]
...
allocated_after_load=1732702
```

Revision c93d07ed7448f332379cf21b4b7b649b057e5671 - 09/02/2024 11:40 PM - yui-knk (Kaneko Yuichiro)

[Bug #20695] Do not create needless string object in parser

set_parser_s_value does nothing in parser therefore no need to
create string object in parser set_yylval_node.

Object allocation

Run ruby benchmarks/lobsters/benchmark.rb with the patch

```
diff --git a/benchmarks/lobsters/benchmark.rb b/benchmarks/lobsters/benchmark.rb
index 240c50c..6cdd0ac 100644
--- a/benchmarks/lobsters/benchmark.rb
+++ b/benchmarks/lobsters/benchmark.rb
@@ -7,6 +7,8 @@ Dir.chdir __dir__
 use_gemfile
```

```
 require_relative 'config/environment'
+printf "allocated_after_load=%d\n", GC.stat(:total_allocated_objects)
+exit
 require_relative "route_generator"

 # For an in-mem DB, we need to load all data on every boot
```

Before

```
ruby 3.4.0dev (2024-08-31T18:30:25Z master d6fc8f3d57) [arm64-darwin21]
...
allocated_after_load=2143519
```

After

```
ruby 3.4.0dev (2024-09-01T00:40:04Z fix_bugs_20695 d1bae52f75) [arm64-darwin21]
...
allocated_after_load=1579662
```

Ruby 3.3.0 for reference

```
ruby 3.3.0 (2023-12-25 revision 5124f9ac75) [arm64-darwin21]
...
allocated_after_load=1732702
```

Revision c93d07ed - 09/02/2024 11:40 PM - yui-knk (Kaneko Yuichiro)

[Bug #20695] Do not create needless string object in parser

set_parser_s_value does nothing in parser therefore no need to
create string object in parser set_yylval_node.

Object allocation

Run ruby benchmarks/lobsters/benchmark.rb with the patch

```
diff --git a/benchmarks/lobsters/benchmark.rb b/benchmarks/lobsters/benchmark.rb
index 240c50c..6cdd0ac 100644
--- a/benchmarks/lobsters/benchmark.rb
+++ b/benchmarks/lobsters/benchmark.rb
@@ -7,6 +7,8 @@ Dir.chdir __dir__
 use_gemfile

 require_relative 'config/environment'
+printf "allocated_after_load=%d\n", GC.stat(:total_allocated_objects)
+exit
 require_relative "route_generator"

 # For an in-mem DB, we need to load all data on every boot
```

Before

```
ruby 3.4.0dev (2024-08-31T18:30:25Z master d6fc8f3d57) [arm64-darwin21]
...
allocated_after_load=2143519
```

After

```
ruby 3.4.0dev (2024-09-01T00:40:04Z fix_bugs_20695 d1bae52f75) [arm64-darwin21]
...
allocated_after_load=1579662
```

Ruby 3.3.0 for reference

```
ruby 3.3.0 (2023-12-25 revision 5124f9ac75) [arm64-darwin21]
...
allocated_after_load=1732702
```

History

#1 - 09/02/2024 11:40 PM - yui-knk (Kaneko Yuichiro)

- Status changed from Open to Closed

Applied in changeset git|c93d07ed7448f332379cf21b4b7b649b057e5671.

[Bug #20695] Do not create needless string object in parser

set_parser_s_value does nothing in parser therefore no need to
create string object in parser set_yylval_node.

Object allocation

Run ruby benchmarks/lobsters/benchmark.rb with the patch

```
diff --git a/benchmarks/lobsters/benchmark.rb b/benchmarks/lobsters/benchmark.rb
index 240c50c..6cdd0ac 100644
--- a/benchmarks/lobsters/benchmark.rb
+++ b/benchmarks/lobsters/benchmark.rb
@@ -7,6 +7,8 @@ Dir.chdir __dir__
 use_gemfile

 require_relative 'config/environment'
+printf "allocated_after_load=%d\n", GC.stat(:total_allocated_objects)
+exit
 require_relative "route_generator"

 # For an in-mem DB, we need to load all data on every boot
```

Before

```
ruby 3.4.0dev (2024-08-31T18:30:25Z master d6fc8f3d57) [arm64-darwin21]
...
allocated_after_load=2143519
```

After

```
ruby 3.4.0dev (2024-09-01T00:40:04Z fix_bugs_20695 d1bae52f75) [arm64-darwin21]
...
allocated_after_load=1579662
```

Ruby 3.3.0 for reference

```
ruby 3.3.0 (2023-12-25 revision 5124f9ac75) [arm64-darwin21]
...
allocated_after_load=1732702
```