Ruby - Bug #7500

Improve GC profiler timings on linux

12/03/2012 11:32 AM - tmm1 (Aman Karmani)

Status:	Closed		
Priority:	Normal		
Assignee:	authorNari (Narihiro Nakamura)		
Target version:	2.0.0		
ruby -v:	ruby 2.0.0dev (2012-12-03 trunk 38149) [x86_64-darwin12.2.0]	**Backport:**	

Description

On linux kernels, getrusage()'s precision depends on the value of HZ when the kernel was compiled. By default, HZ=250 provides a 4ms granularity.

This patch uses clock_gettime() with CLOCK_PROCESS_CPUTIME_ID when available, which provides a 1ns precision on linux.

Associated revisions

Revision ab012bda4a0bdfb720fdf4debbf401f6923f401f - 12/05/2012 02:53 PM - authorNari (Narihiro Nakamura)

- gc.c (getrusage_time): uses clock_gettime() with CLOCK_PROCESS_CPUTIME_ID when available, which provides a 1ns precision on linux. [ruby-core:50495] [Bug #7500]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@38214 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision ab012bda - 12/05/2012 02:53 PM - authorNari (Narihiro Nakamura)

- gc.c (getrusage_time): uses clock_gettime() with CLOCK_PROCESS_CPUTIME_ID when available, which provides a 1ns precision on linux. [ruby-core:50495] [Bug #7500]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@38214 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 12/03/2012 11:46 AM - authorNari (Narihiro Nakamura)

- Category set to core

- Assignee set to authorNari (Narihiro Nakamura)

#2 - 12/03/2012 12:16 PM - kosaki (Motohiro KOSAKI)

AFAIK, clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &ts) return time included kernel running. So, clock_gettime() and ru_utime of getrusage() aren't equivalent.

#3 - 12/03/2012 12:29 PM - kosaki (Motohiro KOSAKI)

And, if i understand the kernel source correctly, getrusage() and get_time(CLOCK_PROCESS_CPUTIME_ID) use the same clock source. So, I doubt this patch improve time accuracy.

Do anyone have a test result?

#4 - 12/04/2012 10:04 AM - tmm1 (Aman Karmani)

tmm1@fe19:~$ uname -a
Linux fe19.rs.github.com 3.5.0-17-generic #28-Ubuntu SMP Tue Oct 9 19:31:23 UTC 2012 x86_64 GNU/Linux

tmm1@fe19:~$ cat timings.c
```
#include <stdlib.h>
#include <stdio.h>
#include <sys/resource.h>
#include <time.h>
#include <math.h>

double getrusage_time() {
```

```
struct rusage usage;
struct timeval time;
getrusage(RUSAGE_SELF, &usage);
time = usage.ru_utime;
return time.tv_sec + time.tv_usec * 1e-6;
}

double clock_time() {
struct timespec ts;

  if (clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &ts) == 0) {
      return ts.tv_sec + ts.tv_nsec * 1e-9;
  }
  return 0.0;

}

int main() {
int n;

  printf("getrusage() before: %f\n", getrusage_time());
  for (n=0; n<10000; n++) pow(2, 2048);
  printf("getrusage() after: %f\n", getrusage_time());

  printf("clock_gettime() before: %f\n", clock_time());
  for (n=0; n<10000; n++) pow(2, 2048);
  printf("clock_gettime() after: %f\n", clock_time());

}

tmm1@fe19:~$ gcc -o timings timings.c -lrt

tmm1@fe19:~$ ./timings
getrusage() before: 0.000000
getrusage() after: 0.000000
clock_gettime() before: 0.001244
clock_gettime() after: 0.001358
```

#5 - 12/04/2012 12:24 PM - tmm1 (Aman Karmani)

Increasing the number of iterations shows the 4ms granularity.

```
tmm1@fe19:~$ grep loops timings.c
int loops = 1000000;
for (n=0; n<loops; n++) pow(2, 2048);
for (n=0; n<loops; n++) pow(2, 2048);

tmm1@fe19:~$ gcc -o timings timings.c -lrt && ./timings
getrusage() before: 0.000000
getrusage() after: 0.004000
clock_gettime() before: 0.006966
clock_gettime() after: 0.010733
```

#6 - 12/05/2012 11:53 PM - authorNari (Narihiro Nakamura)

- Status changed from Open to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r38214.
Aman, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- gc.c (getrusage_time): uses clock_gettime() with CLOCK_PROCESS_CPUTIME_ID when available, which provides a 1ns precision on linux. [ruby-core:50495] [Bug #7500]

Files

gc_timing.patch	586 Bytes	12/03/2012	tmm1 (Aman Karmani)