عنوان: **Microbenchmark**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۰۴/۰۷ ۱۵:۴۳
آدرس: www.dotnettips.info
برچسب‌ها: StringBuilder, Microbenchmark

What Is Micro Benchmark? Micro benchmark is a benchmark designed to measure the performance of a very small and specific piece of code. (^)

البته این موضوع امروزه بیشتر در Java مطرحه تا دات نت (^ و ^ و ^) اما مفاهیم اصلی مختص یک زبان یا پلتفرم نیست.
وقتی در مورد آزمایش بار برای مقایسه کارایی کلاس StrigBuilder تحقیق میکردم به مطلب جالبی برخورد کردم. خلاصش این میشه که برای تست بار قسمتهایی از کدتون میتونین زمان موردنیاز برای اجرای اون کد رو بررسی کنین و چون ممکنه انجام این کار چندین بار نیاز بشه بهتره از متد زیر برای اینکار استفاده کنین:

```
static void Profile(string description, int iterations, Action func) {
    // clean up
    GC.Collect();
    GC.WaitForPendingFinalizers();
    GC.Collect();

    // warm up
    func();

    var watch = Stopwatch.StartNew();
    for (int i = 0; i < iterations; i++) {
        func();
    }
    watch.Stop();
    Console.Write(description);
    Console.WriteLine(" Time Elapsed {0} ms", watch.ElapsedMilliseconds);
}
```

سه خط اول متد بالا برای آماده‌سازی حافظه جهت اجرای تست موردنظر است. برای آشنایی بیشتر با نحوه عملکرد Garbage Collector (^ و ^ و ^) خوندن کتاب فوق العاده CLR via C# - 3rd ed رو پیشنهاد میکنم (فصلهای 21 و 22).
سپس یکبار اکشن موردنظر (که حاوی قطعه کد موردنظره) اجرا میشه تا مسائل مربوطه به بارگذاری‌های اولیه در نتیجه تست تاثیر نزاره (warm up).
در نهایت هم آزمایش بار برای تعداد تکرار درخواست شده انجام میشه و زمان اجرای اون در خروجی چاپ میشه.
برای استفاده از متد فوق میشه از کد زیر استفاده کرد:

```
Profile("a descriptions", how_many_iterations_to_run, () =>
{
   // ... code being profiled
});
```

و برای استفاده از این متد در آزمایش کارایی کلاس StringBuilder میشه از کدی شبیه به کد زیر استفاده کرد:

```
var iterations = 10000000;
var testString = ".NET Tips is awesme!";
do
{
  var sb1 = new StringBuilder(testString);
  var sb2 = new StringBuilder(testString) { Capacity = testString.Length * iterations };
  try
  {
    Profiler.Profile("StringBuilder Profiler", iterations, () => sb1.Append(testString));
    Profiler.Profile("StringBuilder Capacity Profiler", iterations, () => sb2.Append(testString));
  }
  catch (Exception ex)
  {
    Console.WriteLine(ex.Message);
  }
  finally
```

```
  {
    Console.WriteLine("----------------------------------------------------------------");
    sb1.Clear();
    sb2.Clear();
  }
} while (Console.ReadKey(true).Key == ConsoleKey.C); // C = continue
```

البته برای اینکه عملیات مقدار دهی خاصیت Capacity در قسمت warm up متد profile نتایج رو تحت تاثیر قرار نده برای این تست من اون قسمت رو کامنت کردم (اگر این کار رو نکنین زمانهای بدست اومده برای هر دو مورد یکی خواهد بود). اجرای کد بالا نتایج زیر رو تو سیستم من ارائه داد:

```
C:\windows\system32\cmd.exe
StringBuilder Profiler Time Elapsed 581 ms
StringBuilder Capacity Profiler Time Elapsed 238 ms
----------------------------------------------------------------
StringBuilder Profiler Time Elapsed 567 ms
StringBuilder Capacity Profiler Time Elapsed 237 ms
----------------------------------------------------------------
StringBuilder Profiler Time Elapsed 566 ms
StringBuilder Capacity Profiler Time Elapsed 236 ms
----------------------------------------------------------------
StringBuilder Profiler Time Elapsed 566 ms
StringBuilder Capacity Profiler Time Elapsed 237 ms
----------------------------------------------------------------
StringBuilder Profiler Time Elapsed 566 ms
StringBuilder Capacity Profiler Time Elapsed 238 ms
----------------------------------------------------------------
StringBuilder Profiler Time Elapsed 566 ms
StringBuilder Capacity Profiler Time Elapsed 236 ms
----------------------------------------------------------------
StringBuilder Profiler Time Elapsed 563 ms
StringBuilder Capacity Profiler Time Elapsed 236 ms
----------------------------------------------------------------
StringBuilder Profiler Time Elapsed 566 ms
StringBuilder Capacity Profiler Time Elapsed 237 ms
----------------------------------------------------------------
```

می‌بینین که نتایج استفاده از متد موردبحث کمی فرق داره و افزایش کارایی در حالت استفاده از پراپرتی Capacity دیگه حدود 3 برابر نیست و حدود 2 دو برابره. البته زمان بدست اومده برای هر دو مورد نسبت به قبل کاهش داشته که بیشترش میتونه مربوطه به عدم درنظر گرفتن زمان موردنیاز برای ایجاد کلاس StringBuilder در این تست جدید باشه (چون بعید میدونم عملیات پاکسازی حافظه توسط GC تو این تست تاثیر چندانی داشته باشه). درهر حال نتایج این تست بیشتر به واقعیت نزدیکه!

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۷ ۱۷:۴

مطلب جالبی هست از یکی از اعضای تیم کامپایلر سی شارپ :(^)
بحث محاسبه کارآیی در دات نت شامل زمان صرف شده برای JIT اولیه کدها هم هست. به همین جهت اجرای اولیه اندکی بیشتر زمان می‌بره. همچنین GC هم در اینجا در ترد دیگری به موازات کار شما مشغول به کار است و اگر در یک اجرا زمان خوبی بدست آوردید به این معنا نیست که الزاما در اجرای بعدی هم همان زمان را بدست می‌آورید چون GC موکول شده به بعد. ضمن اینکه این نوع محاسبات چون به صورت ایزوله انجام می‌شود عموما بیانگر شرایط دنیای واقعی که پارامترهای زیادی در آن‌ها دخیل هستند، نیست.
و ... اینکه برای خیلی از برنامه نویس‌ها این نوع مقایسه‌ها بیشتر جذاب هستند:
Head-to-head benchmark: C++ vs .NET

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۰۴/۰۷ ۱۹:۸

مطالب شما کاملا صحیح و صادق هست.
اما هدف اینگونه آزمایشات (Microbenchmark) مقایسه قطعات کد درون یک برنامس و مثلا بدست آوردن بهترین روش برای رسیدن به یک هدف مشخص. مثلا همین مثال کلاس StringBuilder که بین دو روش ذکر شده کدام سریعتره و چقدر بهتره و اینکه درنهایت با استفاده از نتایج این آزمایشات و سایر داده‌های موجود کدام روش به صرفه تره. یا مثلا در دستکاری لیست‌های بزرگ استفاده از آرایه به صرفه‌تره یا مثلا یک کالکشن از انواع موجود. در مورد JIT هم با استفاده از بخش warm up سعی شده اثر منفی کامپایل اولیه کد رو در تست از بین ببره (هرچند اثر منفی اجرای خود کد تستر در بار اول همچنان پابرجاس).
اما در مورد اثر GC با اینکه در ابتدای متد مذکور سعی شده تا با پاکسازی اولیه حافظه، سیستم آماده انجام آزمایش بشه ولی هیچ تضمینی نیست که در میانه تست GC بطور خودکار فعال نشه، که میتونه رو نتایج تاثیر منفی بذاره. تو این موارد دیگه خود برنامه نویس باید با درنظر گرفتن این مسئله در مورد نتایج بدست اومده تصمیم بگیره.