УДК 004.051

ИЗМЕРЕНИЕ ВРЕМЕНИ ВЫПОЛНЕНИЯ ПРОГРАММНОГО КОДА В ТЕХНОЛОГИИ .NET

Карчевская М.П., Тархов С.В.

ФГБОУ ВО «Уфимский государственный авиационный технический университет», Уфа, e-mail: karchevskaya.m@mail.ru, ramburger@mail.ru, tarkhov@inbox.ru

Одним из важнейших ресурсов программного продукта является время выполнения его программного кода. Оно, наряду с такими показателями, как функциональные возможности, надежность, практичность, сопровождаемость и мобильность, определяет уровень качества программного продукта. В работе рассмотрен метод измерения времени выполнения программного кода с помощью таймера высокой точности High Precision Event Timer (HPET). Описан класс Stopwatch библиотеки классов FCL каркаса Framework, который работает с HPET и предоставляет удобный набор средств, используемых для измерения времени. Описаны специальные проблемно-ориентированные методы класса Stopwatch технологии .Net в операционной системе Microsoft Windows. Основным преимуществом класса Stopwatch является более точное измерение временных интервалов при наличии таймера высокой точности HPET. В качестве эксперимента исследуется время обработки массивов классов Array, List и ArrayList. Для данных классов исследуется: время формирования элементов массивов; время работы методов Reverse, которые переставляют элементы массивов в обратном порядке; время работы методов Sort, которые сортируют элементы массивов. Приведен программный код на объектно-ориентированном языке программирования C# для платформы .NET Framework.

Ключевые слова: методы класса, Stopwatch, Array, List, ArrayList, C#, эксперимент, временная эффективность алгоритмов, измерение времени программного кода

MEASURING THE EXECUTION TIME OF PROGRAM CODE IN .NET TECHNOLOGY

Karchevskaya M.P., Tarkhov S.V.

Ufa State Aviation Technical University, Ufa, e-mail: karchevskaya.m@mail.ru, ramburger@mail.ru, tarkhov@inbox.ru

One of the most important resources of a software product is the code execution time. It, along with such indicators as functionality, reliability, usability, maintenance and portability, determines the level of quality of a software product. A method for measuring the execution time of a program code using the High Precision High Precision Event Timer (HPET) timer is considered. The Stopwatch class of the FCL framework framework framework, which works with HPET and provides a convenient set of tools used to measure time, is described. Describes special problem-oriented methods of the Stopwatch class of the .Net technology in Windows. The main advantage of the Stopwatch class is a more accurate measurement of time intervals with a high-precision HPET timer. Its main advantage is more accurate measurement of time intervals. As an experiment, the processing time of arrays of classes Array, List and ArrayList is investigated. For these classes, we investigate: the time of the formation of the elements of the arrays; time of work of the Reverse methods, which rearrange the elements of the arrays in the reverse order; the running time of the Sort methods, which sort the elements of the arrays. The program code in the object-oriented programming language C # for the .NET Framework is given.

Keywords: class methods, Stopwatch, Array, List, ArrayList, time efficiency, measurement of program code time

В современной программной инженерии в процессе проектирования, написания и отладки программных продуктов нередко приходится решать вопросы, связанные с определением и последующей оптимизацией времени выполнения программного кода [1]. Не меньший интерес представляют вопросы, связанные с оценкой времени выполнения того или иного кода, написанного на различных языках программирования и на различных компьютерных платформах [2]. Определение времени выполнения программного кода критично:

– для приложений, обрабатывающих данные в режиме реального времени, например приложений, реализующих функции управления техническими устройствами;

– при решении инженерных и научно-технических задач, в которых производятся сложные длительные вычисления, а процессорное время дорого и ограничено, например в задачах проектирования и расчета сложных технических объектов;

– при разработке программ, работающих с аппаратной частью компьютеров, например драйверов для различных устройств;

– для процессов обработки больших данных, требующих значительных временных затрат, например при проведении маркетинговых исследований;

– при генерации страниц в web-приложениях, в которых время критично для пользователя и напрямую связано с производительностью серверов;

– для процессов измерения скорости передачи данных в коммуникационных системах;

– при проведении всевозможных тестирований и др.

В процессе разработки программного обеспечения нередко возникает необходимость отладки фрагментов кода, критичных по времени исполнения, для выявления «узких мест» выполнения программы, потребляющих чрезмерное количество ресурсов [3]. Время выполнения программного кода – это один из важнейших ресурсов, определяющих уровень качества программного продукта, наряду с такими показателями, как функциональные возможности, надежность, практичность, сопровождаемость и мобильность [4]. Оптимизация производительности программного продукта путем сокращения времени выполнения программного кода особенно важна, если он разрабатывается в расчете на длительный жизненный цикл и массовое использование при высоких требованиях к его качеству.

Цель исследования – измерение времени выполнения программного кода с использованием методов класса Stopwatch технологии .Net в ОС Windows для оценки качества программного продукта.

Материалы и методы исследования

Измерение времени выполнения программного кода будем выполнять в ОС Windows (Windows 10 Pro, 64-разрядная), как в одной из самых распространённых в настоящее время операционных систем. В качестве системы программирования используем Microsoft Visual Studio 2013 и язык программирования C#. Описанные ниже возможности измерения времени можно одинаково использовать как в Visual C# .NET, так и Visual Basic .NET, в Visual C++ .NET и Visual J# .NET.

Для проведения измерений применим специальные проблемно-ориентированные методы класса Stopwatch на технологической платформе .Net. Класс Stopwatch основан на HPET (High Precision Event Timer – таймер событий высокой точности). Таймер HPET разработан Microsoft с целью устранения проблем с измерением времени выполнения того или иного программного кода. Частота таймера HPET (минимум 10 МГц) не меняется во время работы системы. Каждая версия OS Windows сама определяет, с помощью каких устройств следует реализовать этот таймер.

Существует несколько способов измерения интервалов времени в ОС Windows. Например, функция timeGetTime возвращает системное время в миллисекундах достаточно точно, однако работает медленно из-за многочисленных промежуточных вызовов и преобразований. Функция GetTickCount работает быстро, но имеет невысокую точность 15 мс (для Windows XP), так как использует прерывания, генерируемые часами реального времени компьютера.

Более точный метод измерения времени в ОС Windows – использование пары функций QueryPerformanceCounter [5] и QueryPerformanceFrequency [6]. Функция QueryPerformanceCounter возвращает текущее значение в тиках, а функция QueryPerformanceFrequency возвращает частоту счетчика производительности. Эти функции работают быстро и имеют высокую точность, так как используют таймер высокой точности High Precision Event Timer (HPET). Временной промежуток между «тиками» этого таймера меньше 1 мс, что позволяет производить достаточно точные измерения [7].

В библиотеке классов FCL каркаса Framework имеется класс Stopwatch [8], который также работает с таймером высокого точности High Precision Event Timer и предоставляет удобный набор средств, используемых для измерения времени. Публичный API класса Stopwatch инкапсулирует следующий набор свойств (табл. 1) и методов (табл. 2).

Таблица 1

Свойства публичного API класса Stopwatch

Название	Описание
Elapsed	Возвращает общее затраченное время, измеренное текущим экземпляром
Elapsed. Milliseconds	Возвращает общее затраченное время в миллисекундах (тип long)
Elapsed. TotalMilliseconds	Возвращает общее затраченное время в миллисекундах (тип double)
ElapsedTicks	Возвращает общее затраченное время в тактах таймера
IsRunning	Возвращает значение, показывающее, запущен ли таймер Stopwatch

Таблица 2

Методы публичного API класса Stopwatch

Название	Описание
Start()	Каждый вызов метода начинает подсчет совокупного затраченного времени для интервала
Stop()	Каждый вызов метода Stop завершает текущий интервал подсчета и фиксирует совокупное затраченное время для интервала
Reset()	Останавливает измерение интервала времени и обнуляет затраченное время
StartNew()	Инициализирует новый экземпляр Stopwatch, задает свойство затраченного времени равным нулю и запускает измерение затраченного времени
Restart()	Останавливает измерение интервала времени, обнуляет затраченное время и начинает измерение затраченного времени

Прежде чем начать измерение затраченного времени с помощью методов класса Stopwatch, необходимо создать экземпляр класса Stopwatch и экземпляр класса TimeSpan (представляет интервал времени) [9]. Затраченное время при помощи свойства Elapsed класса Stopwatch помещается в экземпляр класса TimeSpan. Свойства TotalSeconds и TotalMilliseconds экземпляра TimeSpan, возвратит время в секундах или миллисекундах. Чтобы очистить совокупное затраченное время в существующем экземпляре Stopwatch, необходимо использовать метод Reset.

Результаты исследования и их обсуждение

Использование методов класса Stopwatch рассмотрим на примере исследования времени работы программного кода при обработке массивов классов Array, ArrayList и List [10, 11], содержащих *n* строковых данных. Для данных классов вычисляется время:

– формирования элементов массивов;
– работы методов Reverse, переставляют элементы массивов в обратном порядке;
– время работы методов Sort, которые сортируют элементы массивов.

Программный код исследования, написанный на C# в Visul Studio 2013 [12]:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Diagnostics;
using System.Collections;
namespace Снятие_времени
{ public partial class Form3 : Form
  { public Form3()
    {InitializeComponent();}
    private void Form3_Load(object sender, EventArgs e)
     { }
     private void button1_Click(object sender, EventArgs e)
       {Stopwatch d = new Stopwatch();// Создание экземпляра класса Stopwatch
        TimeSpan time = new TimeSpan();//Создание экземпляра класса TimeSpan
        Random f = new Random();
        Random r = new Random();
        System.Text.StringBuilder s;
        int n;  n = Int32.Parse(textBox10.Text);
        //Формирование массива Array строковых данных
        d.Reset(); //Обнуление времени
        d.Start(); // Запуск измерения времени
       string[] A = new string[n]; //массив класса Array, состоящий из n строковых данных
        for (int i = 0; i < n; i++)   //в цикле формируются n строк
```

```
    {
   int k; k = r.Next(3, 10);  // длина формируемой строки задается случайным числом
      s = new System.Text.StringBuilder(5);
      for (int j = 1; j <= 5; j++) s = s.Append(Convert.ToChar(f.Next(65, 90)));
       A[i] = Convert.ToString(s);
    }
    d.Stop();
    time = d.Elapsed;
    textBox1.Text = time.TotalSeconds.ToString("0.000000");
    //Формирование  элементов списка ArrayList  строковых данных
    d.Reset();  d.Start();
    ArrayList A_L = new ArrayList(n); //список ArrayList
    for (int i = 0; i < n; i++)
    {
       s = new System.Text.StringBuilder(5);
       for (int j = 1; j <= 5; j++)  s = s.Append(Convert.ToChar(f.Next(65, 90)));
       A_L.Add(Convert.ToString(s));
    }
    d.Stop();  time = d.Elapsed;
    textBox2.Text = time.TotalSeconds.ToString("0.000000");
    //Формирование  элементов списка List строковых данных
    List<string> L = new List<string>(n); //Список List
    for (int i = 0; i < n; i++)
    {
       s = new System.Text.StringBuilder(5);
       for (int j = 1; j <= 5; j++) s = s.Append(Convert.ToChar(f.Next(65, 90)));
       L.Add(Convert.ToString(s));
    }
    d.Stop(); time = d.Elapsed;
    textBox3.Text = time.TotalSeconds.ToString("0.000000");
    //Перестановка элементов массива в обратном порядке (метод Reverse)
    d.Reset(); d.Start();
    Array.Reverse(A);
    d.Stop(); time = d.Elapsed;
    textBox4.Text = time.TotalSeconds.ToString("0.000000");
    //Перестановка элементов списка ArrayList в обратном порядке (метод Reverse)
    d.Reset(); d.Start();
    A_L.Reverse();
    d.Stop(); time = d.Elapsed;
    textBox5.Text = time.TotalSeconds.ToString("0.000000");
     //Перестановка элементов списка List в обратном порядке (метод Reverse)
    d.Reset(); d.Start();
    L.Reverse();
    d.Stop(); time = d.Elapsed;
    textBox6.Text = time.TotalSeconds.ToString("0.000000");
    //Сортировка Sort n элементов массива строковых данных
    d.Reset(); d.Start();
    Array.Sort(A);
    d.Stop();  time = d.Elapsed;
    textBox7.Text = time.TotalSeconds.ToString("0.000000");
    //Сортировка Sort элементов списка ArrayList
    d.Reset();  d.Start();
    A_L.Sort();
    d.Stop(); time = d.Elapsed;
    textBox8.Text = time.TotalSeconds.ToString("0.000000");
    //Сортировка Sort элементов списка List строковых данных
    d.Reset(); d.Start();
    L.Sort();
    d.Stop(); time = d.Elapsed;
    textBox9.Text = time.TotalSeconds.ToString("0.000000");
}}}
```

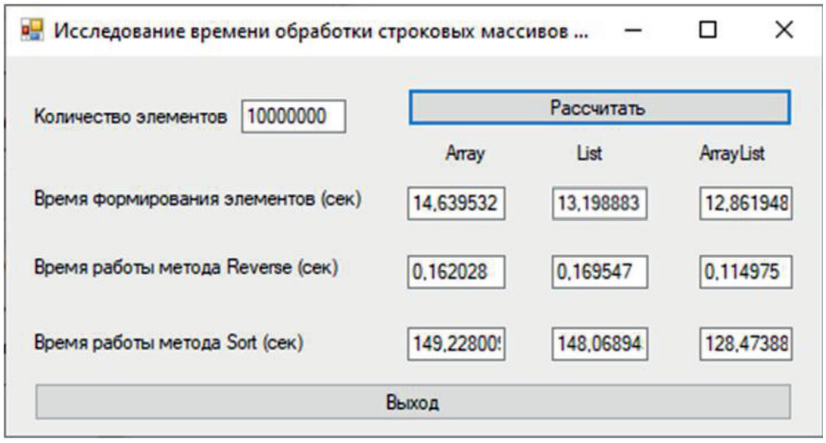

Рис. 1. Результат исследования времени работы программного кода при обработке массивов классов Array, List и ArrayList

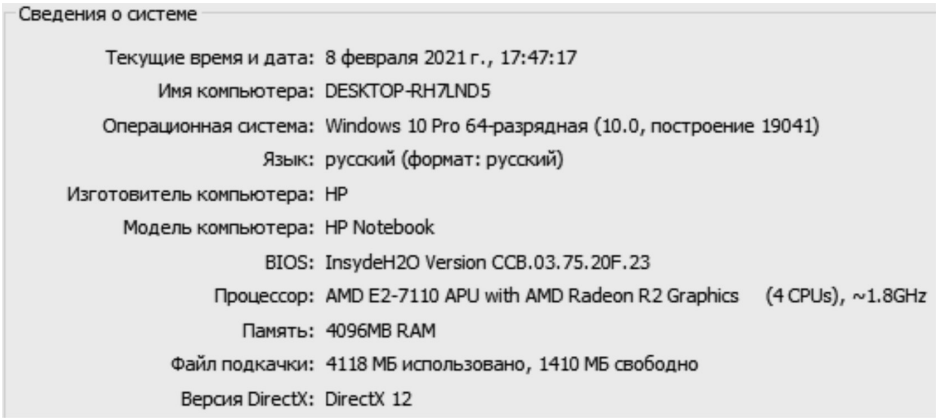

Рис. 2. Характеристика компьютера и ОС, на котором проводилось исследование

Поскольку ОС Windows – многозадачная операционная система, то получить «чистое» время выполнения программного кода невозможно, оно всегда будет относительным. Делать выводы о времени выполнения какого-либо алгоритма после одного программного тестирования нельзя, поэтому алгоритм был протестирован 20 раз и затем вычислено среднее значение времени. Результат эксперимента приведен на рис. 1.

Характеристики компьютера, на котором проводился эксперимент по исследованию времени работы программного кода при обработке массивов классов Array, List и ArrayList, приведены на рис. 2.

Заключение

Операционная система Windows позволяет измерять интервалы времени с использованием различных таймеров (функций). Функция timeGetTime дает достаточно точный результат, но работает медленно из-за многочисленных промежуточных вызовов и преобразований. Функция GetTickCount работает быстро, но имеет невысокую точность. Наиболее предпочтительным для практического применения при измерении времени выполнения программного кода является таймер высокой точности High Precision Event Timer (HPET), сочетающий преимущества функций timeGetTime и GetTickCount. Таймер высокой точности HPET является одновременно точным и быстрым. Он использует функции QueryPerformanceCounter и QueryPerformanceFrequency.

Проведенный с использованием таймера высокой точности HPET эксперимент на компьютере с операционной системой Windows 10 Pro (64-разрядная ОС, процессор: AMD E2-7110 APU with AMD Radeon R2 Graphics (4 CPUs), 1.8GHz) позволил провести измерение времени, затраченного на обработку строковых данных с помо-

щью методов класса Stopwatch. При этом выполнялись операции с массивами класса ArrayList, Array и List. Результаты измерений показали, что операции с массивами класса ArrayList выполняются быстрее, чем операции с массивами классов Array и List. Использование классов Stopwatch, Array, List и ArrayList в других языках программирования, входящими в Visual Studio, будет отличаться лишь особенностями конкретного языка [13].

Класс Stopwatch библиотеки классов FCL каркаса Framework, который работает с HPET и предоставляет удобный набор средств, используемых для измерения времени. Основным его преимуществом является более точное измерение временных интервалов. Однако следует отметить, что при отсутствии таймера высокой точности HPET его преимущества теряются.

Список литературы

1. Лохматов С.Ю., Светлов С.В., Калач Г.П. Методы измерения времени работы программы // Современные научные исследования и инновации. 2017. № 5 [Электронный ресурс]. URL: http://web.snauka.ru/issues/2017/05/83188 (дата обращения: 11.03.2021).

2. Хашковскии В.В., Лутай В.Н., Юрченко В.В. Сравнительная оценка времени выполнения программ на различных платформах / Известия ЮФУ. Техн. науки. 2009. С. 176–180.

3. Ефремов М.А., Кирьянчиков В.А. Разработка методики и программного средства для измерения времени выполнения фрагментов программного кода // Современное образование: содержание, технологии, качество. СПб, «ЛЭТИ» им. В.И. Ульянова. 2019. Т. 1. С. 403–405.

4. ГОСТ ИСО/МЭК 9126-2001 «Информационные технологии. Оценка программной продукции. Характеристики качества и руководства по их применению». Минск: Евразийский совет по стандартизации, метрологи и сертификации, 2006. 13 с.

5. Библиотека документации для разработчиков под ОС MS Windows. [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/windows/desktop/ms644904(v=vs.85).aspx (дата обращения: 11.03.2021).

6. Библиотека документации для разработчиков под ОС MS Windows. [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/windows/desktop/ms644905(v=vs.85).aspx (дата обращения: 11.03.2021).

7. Карчевская М.П., Рамбургер О.Л. Измерение времени выполнения программного кода в ОС Windows с использованием ИСР Lazarus // Задачи обработки больших данных в авиации: материалы II Всероссийской научно-практической конференции «Свободный полет – 2015». Жуковский. Уфа, 2016. С. 127–133.

8. Библиотека документации для разработчиков под ОС MS Windows. [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/system.diagnostics.stopwatch(v=vs.110).aspx (дата обращения: 11.03.2021).

9. Библиотека документации для разработчиков под ОС MS Windows. [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/system.timespan(v=vs.110).aspx (дата обращения: 11.03.2021).

10. Библиотека документации для разработчиков под ОС MS Windows. [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/system.array(v=vs.110).aspx (дата обращения: 11.03.2021).

11. Библиотека документации для разработчиков под ОС MS Windows. [Электронный ресурс]. URL: https://msdn.microsoft.com/ru-ru/library/system.collections.arraylist(v=vs.110).aspx (дата обращения: 11.03.2021).

12. Карчевская М.П., Кузьмина Е.А., Рамбургер О.Л., Смирнова Е.А. Технология программирования на C#: учебное пособие. Уфа: РИК УГАТУ, 2016. 278 с.

13. Карчевская М.П., Рамбургер О.Л. Технология программирования на VB.NET: учебное пособие. Уфа: УГАТУ, 2014. 225 с.