УДК: 004.074.32
DOI: 10.30987/article_5c387d60657774.94658729

Г.А. Асланов, Л.Ф. Сугарова

ИССЛЕДОВАНИЕ ФАКТОРОВ, ВЛИЯЮЩИХ НА СКОРОСТЬ ВЫЗОВА ПРОЦЕДУР ЯЗЫКА С++

Представлены результаты исследования, доказывающие, что время на передачу управления функции и возврат из нее при использовании методов макрозамен не влияют на скорость работы программы, написанной на С++.

Ключевые слова: *программный код, оптимизация, модель, качество, производительность.*

G.A. Aslanov, L.F. Sugarova

STUDY OF FACTORS INFLUENCING THE SPEED OF CALLING PROCEDURES, LANGUAGE C++

The results of the research are presented, which prove that the time spent on transferring control of a function and returning from it when using methods of macro-substitutions does not affect the speed of the program written in C ++.

Keywords: *program code, optimization, model, quality, output.*

Введение

Целью проведенного исследования является: определение факторов, влияющих на скорость вызова функции в языке С++.

Значение прироста в производительности в результате макрозамен включает два слагаемых: 1) время на передачу управления в функцию и возврат из неё; 2) время на выделение стековой памяти функции, необходимой для размещения всех локальных переменных и массивов, - и может быть представлено в виде выражения (1):

$$T = t^{call} + t^{stack} \quad (1)$$

t^{call} – время на передачу управления в функцию и возврат из нее без учета времени, необходимого на выделение локального стека функции. Эта величина является постоянной и соответствует времени выполнения машинной инструкции CALL(передающей управление процедуре на языке Ассемблера).

t^{stack} – время, затрачиваемой на выделение локальной (стековой) памяти, предназначенной для размещения переменных, объявленных внутри блока функции. Это время зависит от суммарного размера локальных данных, ниже данное утверждение будет подтверждено экспериментально.

Чтобы оценить порядок величин t^{call}, t^{stack} и оценить их значимость в выражении (1) были проведены ряд экспериментов. Экспериментальные замеры проводились с помощью тестового кода, написанного на языке С++ в среде Visual Studio. Для получения максимально объективных оценок в настройках проекта параметру «Optimization» было присвоено значение «Disabled (/Od)», что предотвращает влияние встроенных в компилятор Microsoft методов оптимизации. Кроме того, параметр «Inline function expansion» был установлен в

«Only __inline (/Ob1)» - данный флаг запрещает компилятору игнорировать инструкцию inline при построении исполняемого кода (рис.1). Замеры проводились в режиме «Release» для исключения диагностического кода.

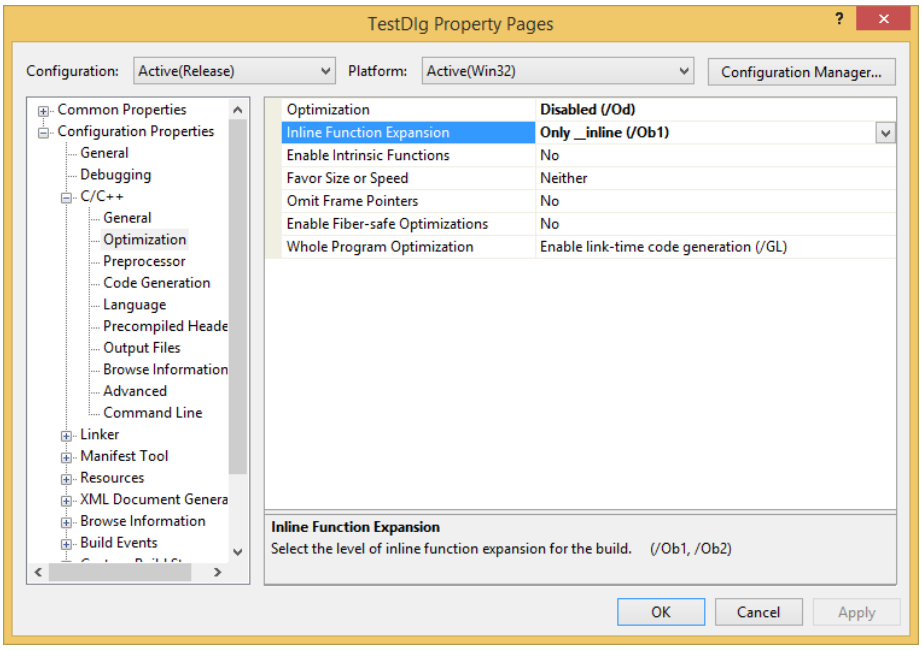

Рис. 1. Настройки тестового проекта

1. Процесс тестирования

Для оценки величины времени вызова и возврата из функции t^{call} был проведен тест, в котором замерялось время выполнения кода, выполняющего в цикле функцию с пустым телом и без параметров (чтобы исключить влияние операторов выделения стека). Количество итераций цикла последовательно изменялось от 1000000 до 20000000 с шагом 1000000. Исходный код теста имеет вид:

```
CString cstr="";
for (int N=1000000; N<=20000000; N+=1000000){
        DWORD dwStart = GetTickCount();
        for (int i=0; i<N; i++){
                f();
        }
        DWORD dwTime = GetTickCount() - dwStart;
        CString cs; cs.Format("%u\n",dwTime);
        cstr += cs;
}
AfxMessageBox(cstr);
```

Были проведены два замера: в первом функция была объявлена обычным образом:

```
void f()
{

}
```

Во втором случае с использованием инструкции _forceinline:

```
__forceinline void f()
{

}
```

Результаты замеров показали, что отличия во времени работы лежали в пределах статистической погрешности. Максимальное время выполнения (для 20000000 итераций) составило около 47 микросекунд.

Так как существовала возможность того, что компилятор игнорирует флаг запрета встроенной оптимизации и все-таки исключает функции с пустым телом из объектного кода, то были проведены 2 дополнительных теста: в первом вместо функции f() в цикле вызывалось выражение *log(10.5)*:

```
        CString cstr="";
 for (int N=1000000; N<=20000000; N+=1000000){
          DWORD dwStart = GetTickCount();
          for (int i=0; i<N; i++){
                  log(10.5);
          }
          DWORD dwTime = GetTickCount() - dwStart;
          CString cs; cs.Format("%u\n",dwTime);
          cstr += cs;
 }
 AfxMessageBox(cstr);
```

А во втором тесте выражение *log(10.5);* было помещено в тело функции *f()*:

```
void f()
{
  log(10.5);
}
```

Сравнение результатов этих двух тестов также показало очень незначительные расхождение. Это говорит о том, что затраты на вызов функции и возврат из неё незначительны. Выражение (1) примет следующий вид:

$$T = t^{stack} \tag{2}$$

Для оценки зависимости времени выделения стековой памяти от размера локальных данных был проведен следующий эксперимент: число итераций было зафиксировано значением 20000000, Далее были проведены 10 замеров для каждого из которых менялся размер массива «y», объявленного в теле функции f() (Листинг 1) от 100000 до 1000000 байт включительно:

```
void f()
{
  char y[1000000];
  y[0] = 'a';
}
```

2. Результаты экспериментов

Аналогичный эксперимент был проведен для варианта функции f() с использованием модификатора __forceinline. Результаты обоих экспериментов приведены в таблице 1.

Полученные результаты говорят о высокой эффективности использования модификатора __forceinline: очевидно, что объявление данных компилятор поместил перед циклом. На графике, изображенном на рисунке 2 хорошо виден линейный характер зависимости времени выделения локального стека функции от его размера: поверх кривой экспериментальных данных, взятых из первой и второй колонок таблицы 1, проведена прямая (пунктиром), коэффициенты, которой получены аппроксимацией данных методом МНК.

Таблица 1 - Результаты замеров времени выделения локальной памяти

Размер данных	время работы обычной функции	время работы __forceinline функции
100000	579	47
200000	1312	47
300000	2187	47
400000	3468	47
500000	4343	47
600000	5187	47
700000	6079	47
800000	6922	47
900000	7781	47
1000000	8641	47

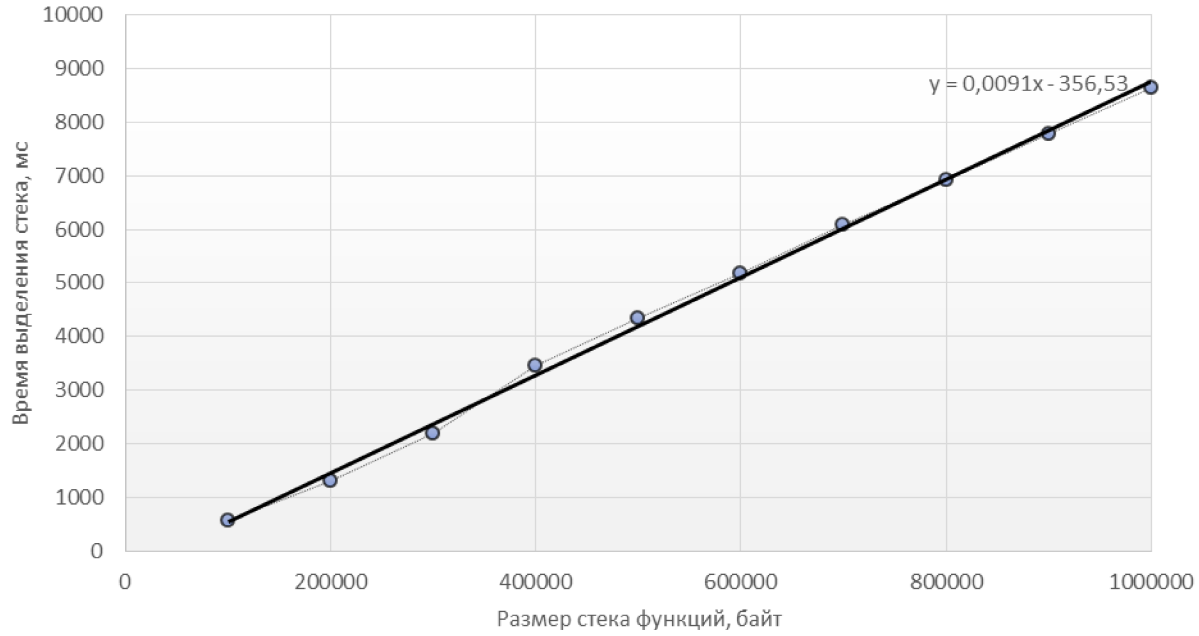

Рис. 2. Кривая экспериментальных данных и прямая, описывающая линейную зависимость времени выделения стековой памяти от её размера.

На графике, изображенном на рисунке 2 ось абсцисс, соответствует размеру стека (в байтах), а ось ординат – времени выделения стека (в миллисекундах).

Проведенные эксперименты показали, что в выражение (1) можно упростить, убрав из него представив выигрыш от макрозамены t^{call}, ввиду его фактической несущественности. Кроме того, подтвержденный экспериментально линейный характер зависимости времени выделения стека от его размера позволяет с высокой степенью точности использовать для прогнозирования времени выделения стека коэффициенты пропорциональности либо более наглядный показатель скорости выделения стека, с учетом этого выражение (1) можно преобразовать в следующее (2):

$$T = \frac{v}{s} \tag{2}$$

v – суммарный размер локальных переменных (стека функции);
s – скорость выделения стека.

Выводы

В методе макрозамен, как и во всех задачах, относящихся к категории моделей «экстремального программирования», улучшение качества программы достигается за счет использования дополнительных вычислительных ресурсов. Макрозамены приводят к увеличению размера кода программы пропорционально количеству вызовов функций, для которых такая замена будет осуществляться. Естественно, в сложных системах с большим числом функций, неограниченное использование макрозамен невозможно – прежде всего из-за того, что оперативная память, используемая для размещения исполняемого кода, является ресурсом достаточно дорогим. Таким образом имеет место оптимизационная проблема выбора стратегии макрозамен, улучшающей производительность кода, в условиях ограниченного объема ресурсов оперативной памяти, необходимого для достижения данной цели.

Список литературы:

1. Томаев М.Х., Асланов Г.А., Ванюшенкова Н.В. Использование оптимизационных моделей экстремального программирования в проектировании ПО // IT-Технологии: теория и практика, Материалы семинара, 2017.
2. Соколова Е.А., Определение параметров быстродействия алгоритма компрессии статичных изображений // Междунар. науч. конф. молодых ученых, студентов и аспирантов «Перспектива-2008»: сб. матер. Нальчик: Каб.-Балк. ун-т, 2008. С.143-147.
3. Соколова Е.А. Математическая модель компрессии статичных изображений вариабельными фрагментами с учетом погрешностей // деп. в ВИНИТИ 19.07.07. № 748-В2007, указатель № 9, 12 с.
4. Соколова Е.А. К проблеме повышения эффективности компрессии изображений // Безопасность информационных технологий, Министерство образования и науки РФ, МИФИ, ВНИИПТИ. 2008. № 2. С.57-60.
5. Соколова Е.А. Разработка метода сохранения пикселей в массив с дифференциацией на цветовые компоненты // Международный научно-исследовательский журнал. 2017. № 7 (61). С. 7.
6. Соколова Е.А. Метод компрессии цифровых трехмерных изображений с помощью анализа таблицы текстурных координат // Международный научно-исследовательский журнал. 2017. № 7 (61). С. 9.

References:

1. Tomaev M.KH., Aslanov G.A., Vanyushenkova N.V. (2017). Use of optimization models of extreme programming in software design. IT-Technologies: theory and practice, Seminar materials. [in Russian language]
2. Sokolova E.A. (2008). Determination of the parameters of the speed of the compression algorithm for static images. International scientific conference young scientists, students and post-graduate students «Perspective-2008»: Cal. mater. Nalchik: Cab-Balk. University, pp.143-147. [in Russian language]
3. Sokolova E.A. (2007) Mathematical model of compression of static images with variable fragments taking into account errors. Dep. in VINITI No. 748-B2007, the index No. 9. [in Russian language]
4. Sokolova E.A. (2008). To the problem of increasing the efficiency of image compression. Information Technology Security, Ministry of Education and Science of the Russian Federation, MEPhI, VNIIPTI, (2), pp. 57-60. [in Russian language]
5. Sokolova E.A. (2017). Development of a method for storing pixels in an array with differentiation into color components. International Scientific and Research Journal, 61(7). [in Russian language]
6. Sokolova E.A. (2017). The method of compression of digital three-dimensional images using the analysis of the table of texture coordinates. International Scientific and Research Journal, 61(7). [in Russian language]

Статья поступила в редколлегию 31.09.18.
Рецензент: к.т.н., доцент Брянского государственного технического университета Подвесовский А.Г.
Статья принята к публикации 12.12.18.

Сведения об авторах

Асланов Геворк Артурович
магистр, ФГБОУ ВО «Северо-Кавказский горно-металлургический институт (ГТУ)»,
тел.: +7 (962) 747-88-47
E-mail: gevork5@mail.ru

Сугарова Лаура Феликсовна
ГБОУ "Гимназия "Диалог", 11-а класс,
тел.: +7 (918) 828-71-24

Information about authors:

Aslanov Gevorg Arturovich
Master, FSBEI HE "North Caucasus Mining and Metallurgical Institute (State Technological University) "
tel.: +7 (962) 747-88-47
E-mail: gevork5@mail.ru

Sugarova Laura Feliksovna
GBOU "High School"Dialogue", 11th grade,
phone: +7 (918) 828-71-24