DOI 10.37882/2223–2966.2022.10.32

ЗАДАЧА ОПТИМИЗАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ РАЗМЕЩЕНИЯ МАССИВОВ ДАННЫХ ПЕРЕМЕННОЙ ДЛИНЫ В ОПЕРАТИВНОЙ ПАМЯТИ ЭВМ

THE PROBLEM OF OPTIMIZING THE PERFORMANCE OF PLACEMENT OF VARIABLE-LENGTH DATA ARRAYS IN COMPUTER RAM

M. Tomaev
A. Sanakoev

Summary. A method is proposed for optimizing the performance of computer RAM allocation in software systems for which the sizes of information arrays change with equal probability in known intervals. A formulation and an effective algorithm for solving the problem are proposed.

Keywords: program, memory, system, model, performance, efficiency, optimization, algorithm, array.

Томаев Мурат Хасанбекович
Кандидат технических наук, доцент
Северо-кавказский государственный горно-металлургического институт (государственный технологический университет)
Владикавказ
tmxwork@mail.ru
Санакоев Алибек Викторович
Магистрант
Северо-кавказский государственный горно-металлургического институт (государственный технологический университет)
Владикавказ
sanakoev.alibek@gmail.com

Аннотация. Предлагается метод оптимизации производительности выделения оперативной памяти ЭВМ в программных системах, для которых размеры информационных массивов равновероятно меняются в известных интервалах. Описываются формулировка и эффективный алгоритм решения задачи.

Ключевые слова: программа, память, система, модель, производительность, эффективность, оптимизация, алгоритм, массив.

1. Введение

В прикладных программных алгоритмах, характеризующихся требовательностью к объему доступной оперативной памяти [1,2], эффективное управление размещением данных может дать значительный прирост производительности. Одним из способов улучшить производительность прикладного кода [3] может стать перемещение массивов данных в область памяти с более низкими накладными расходами. Чаще всего для размещения данных используют динамическую память, которая склонна к дефрагментации. Один из способов перемещения из динамической кучи в статическую память демонстрируются в следующих двух листингах исходного кода на языке C++:

Листинг 1.
Размещение массива в динамической куче

```
double *arr = new double [RequriedSize];
```

Листинг 2. Альтернативный код, позволяющий размещать данные статической (глобальной области) памяти, если заданный лимит «ReservedSize» не превышен

```
static double staticbuffer_for_arr[ReservedSize];
double *arr = (RequriedSize > ReservedSize?
new double[RequriedSize]: staticbuffer_for_arr);
```

В случае использования оптимизации, описанной в Листинге 2, освобождение блока, занятого массивом, также требует изменений:

Листинг 3.
Модифицированный код освобождения

```
if (staticbuffer_for_arr!= arr) delete [] arr;
```

Очевидно, что использование вспомогательных статических массивов, подобных «staticbuffer_for_arr»,

расширит размер постоянной области ОП, требуемый приложению при запуске. Так как оперативная память является очень ценным ресурсом, то актуальным является задача его оптимального распределения с целью достижения наилучшего качества кода. Применительно к данному методу в критерий качества [4,5] формулируется как суммарное время выделения динамических блоков памяти.

В следующей главе предлагается способ распределения постоянной памяти для программных алгоритмов с предсказуемыми интервалами равновероятного изменения размера динамических массивов при известной верхней границе доступного объема оперативной памяти.

2. Формальная постановка задачи

2.1. Обозначения

Пусть для каждого динамического массива прикладного алгоритма известны диапазон изменения размера $[A_i,B_i]$ (причем любое значение в пределах этого интервала — величина равновероятная), а также статистически достоверное количество операций резервирования памяти C_i (данную величину можно заменить нормированной величиной относительной частоты). Тогда суммарное среднее время выделения памяти в динамической куче для всех массивов будет равно:

$$F_1 = \sum_{i=1}^{N} C_i \left(\frac{A_i + B_i}{2}\right) / s \quad (1)$$

где

s — среднее скорость резервирования динамической памяти.

Использование оптимизационного подхода изменит нижнюю границу интервала обращений к динамической памяти — она станет равна размеру вспомогательного статического массива x_i, таким образом второй множитель

$$\left(\frac{A_i + B_i}{2}\right)$$

выражения (1), представляющий собой средний размер используемых динамических блоков памяти, в этом случае примет вид:

$$\left(\frac{x_i + B_i}{2}\right) \quad (2)$$

Аналогично, уменьшение исходного интервала с $[A_i,B_i]$ на $[ReservedSize,B_i]$ приведет к пропорциональному изменению количества обращений к менеджеру динамической памяти, т.е. первый множитель C_i выражения (1) изменится на:

$$C_i \frac{B_i - x_i}{B_i - A_i} \quad (3)$$

Окончательно, с учетом известного лимита доступной (верхней границы) оперативной памяти V, задачу оптимального распределения статической памяти между динамическими массивами можно представить в виде следующей непрерывной оптимизационной модели (4):

$$\begin{cases} F_2 = \sum_{i=1}^{N} \left(C_i \frac{B_i - x_i}{B_i - A_i}\right) \frac{B_i + x_i}{2s} \to min; \\ \sum_{i=1}^{N} ReservedSize_i \le V; \\ ReservedSize_i = signum(x_i - A_i) x_i; \\ \forall_i : A_i \le x_i \le B_i. \end{cases} \quad (4)$$

Здесь

V — значение верхней границы оперативной памяти, доступной для оптимизации;

$ReservedSize_i$ — уточненное значение размера вспомогательного статического массива, используемого для размещения i-го массива данных.

Введение дополнительного обозначения $ReservedSize_i$ потребовалось для «отсечения» случая, когда размер вспомогательного массива x_i выбирается равным нижней границе A_i.

Введя два вспомогательных обозначения (5) и (6):

$$D = \sum_{i=1}^{N} C_i \frac{B_i^2}{2s(B_i - A_i)} \quad (5)$$

$$g_i = \frac{C_i}{2s(B_i - A_i)} \quad (6)$$

Можно представить задачу (1) в новом виде (7), более наглядно представляющем влияние отдельных составляющих выражения целевой функции на её значение:

$$\begin{cases} F_3 = D - \sum_{i=1}^{N} g_i x_i^2 \to min; \\ \sum_{i=1}^{N} ReservedSize_i \le V; \\ ReservedSize_i = signum(x_i - A_i) x_i; \\ \forall_i : A_i \le x_i \le B_i. \end{cases} \quad (7)$$

Результатом преобразований (5), (6), (7) явилось выделение неизвестной x_i в составе множителя x_i^2. Учитывая характер выражений целевой функций и ограничения, очевидно, что решением задачи (7) является последовательное распределение между массивами объема доступной оперативной памяти V в порядке

убывания коэффициента g_i. Следует отметить, что скорость выделения динамической памяти «s», используемая в выражениях (5), (6) не оказывает влияние на выбор оптимальных значений $ReservedSize_i$, поэтому её можно принять равной единице. Полученный в результате массив элементов $ReservedSize_i$ содержит значение вспомогательного статического массива, который рекомендуется к использованию для i-го массива — в этом случае к массиву можно применить оптимизацию, описанную в *Листинге 2* предыдущего раздела. Значение $ReservedSize_i$, равное нулю, означает рекомендацию не использовать оптимизацию для i-го массива, т.е. код следует оставить в исходном виде (*Листинг 1*).

3. Заключение

Актуальность подхода к формулировке частной задачи оптимизации производительности, предложенного в работе, заключается в отходе от попыток определения точных оценок загруженности того или иного участка кода, в пользу интервального подхода, когда задается достоверный диапазон изменения нагрузки — такой метод позволяет более точно описывать характер большинства программных систем. Важным результатом является наглядная демонстрация непрерывного характера задачи и быстрый алгоритм решения. Использование результатов работы возможно в составе оптимизирующих трансляторов либо автоматизированных систем проектирования ПО.

ЛИТЕРАТУРА

1. Гроппен В.О., Томаев М.Х. Модели, алгоритмы и средства программной поддержки проектирования оптимальных программных продуктов. Автоматика и телемеханика. ИПУ РАН. г. Москва, 2000, вып. 11., стр. 175–184.
2. Томаев М.Х., Губиев Д.А. Средства автоматизации оптимизационных преобразований исходных кодов программных систем. Электронный научный журнал: Программные продукты системы и алгоритмы, 3 выпуск, 2018., http://swsys-web.ru/ru/means-of-automation-of-transformations-of-source-codes-of-program-systems.html.
3. Босиков И.И., Томаев М.Х., Гамиди А.О. Формализация метода кэширования функций произвольного числа переменных. Наука и бизнес: пути развития № 11(101) 2019, с. 75.
4. Томаев М.Х. Использование оптимизационных моделей «экстремального программирования» в проектировании ПО. Выбор оптимальной стратегии макрозамен.IT-технологии: теория и практика. Материалы семинара. Владикавказ, 2017, стр. 39–55.
5. Томаев М.Х., Миронян Р.А. Формализации метода статического кэширования функций. Материалы международной научно-технической конференции «ИТ-технологии: развитие и приложения». Владикавказ, 2019, с. 45.

Журнал «Современная наука: актуальные проблемы теории и практики»