Модел. и анализ информ. систем. Т. **17**, № **2** (2010) 122–132

УДК 519.6

Метод оценки частоты выполнения фрагментов кода последовательной программы

Шалимов А.В.

МГУ им. М.В. Ломоносова

e-mail: ashalimov@lvk.cs.msu.su

получена 1 ноября 2009

Ключевые слова: частота выполнения, анализ программ, метод статистических испытаний, профилирование.

Рассматривается задача о вычислении частоты выполнения фрагментов кода последовательной программы. Эта задача встречается во многих приложениях: оптимизация программ, распараллеливание программ, распределение ресурсов вычислителя, компактное представление программ в памяти, выявление вредоносного программного обеспечения. В статье предложен новый метод оценки частоты выполнения линейных участков программы на основе метода статистических испытаний, позволяющий с заданной точностью оценить число испытаний программы.

1. Введение

В статье рассматривается задача вычисления частоты выполнения фрагментов кода последовательной программы. Эта задача возникает во многих приложениях.

- Оптимизация, распараллеливание и тестирование программ: для того чтобы сконцентрировать усилия именно на активных участках программ.
- Повышение эффективности алгоритмов управления ресурсами в операционных системах.
- Выбор резидентной части в системах реального времени.
- Управление нагрузками в распределенных и многопроцессорных системах.

Под фрагментом кода понимается линейный участок программы, то есть такой участок кода, где не нарушается естественный порядок выполнения.

Данная задача имеет тесную связь с областью профилировки программ [3, 4], где возникает задача определения частоты выполнения инструкций программы. Существующие решения в этой области сводятся к применению различных эмпирических данных или к предположению, что все входные данные программы равномерно распределены на области значений, что в общем случае не всегда верно.

2. Описание задачи

Пусть $\Pi(x_1, \ldots, x_p) = \{V, E\}$ – исходная последовательная программа с p входными параметрами $(x_1, \ldots, x_p)$. Программа представлена в виде графа потока управления с вершинами $V = \{b_j\}$, где b_j – линейные участки программы, $j = \overline{1, m}$, и дугами $E = \{(b_{j_1}, b_{j_2})\}$, где (b_{j_1}, b_{j_2}) – переход управления от j_1 линейного участка к j_2 линейному участку.

Для каждого входного параметра $x_1, \ldots, x_p$ известно конечное множество допустимых значений и функция распределения этих значений. Программа $\Pi(x_1, \ldots, x_p)$ не зацикливается на допустимых наборах входных данных (каждый линейный участок выполняется конечное число раз).

Введем следующие обозначения.

- $T(x_i)$ – множество допустимых значений входного параметра x_i;
- $\hat{x}_i$ – значение входного параметра из $T(x_i)$;
- $\Pi_j(\hat{x}_1, \ldots, \hat{x}_p)$ – число выполнений j-го линейного участка на векторе входных значений $(\hat{x}_1, \ldots, \hat{x}_p)$;
- $M_p = T(x_1) \times \cdots \times T(x_p)$ – множество всех наборов входных данных мощности $|M_p|$ и размерности p.

Будем рассматривать каждый входной параметр программы как случайную величину с заданной функцией распределения значений [1, 2]. Сразу отметим, что на практике входным параметрам большинства программ можно сопоставить некоторую случайную величину. Например, входному параметру, отвечающему за показания альтиметра (пилотажно-навигационный прибор, указывающий высоту полета). Погрешности показаний альтиметра обусловлены суммарным воздействием большого числа случайных факторов. Тогда, в силу центральной предельной теоремы, показания альтиметра могут считаться нормально распределенными с дисперсией, обусловленной точностью прибора.

Итак, пусть $X_1, \ldots, X_p$ – случайные величины для входных параметров $(x_1, \ldots, x_p)$. Тогда $Y_j = \Pi_j(X_1, \ldots, X_p)$ является случайной величиной для числа выполнений j-го линейного участка. Отметим, что Y_j – это дискретная случайная величина, так как число выполнений линейного участка при одном испытании принадлежит множеству натуральных чисел с нулем $N \cup \{0\}$ и принимает только конечные значения.

Определение 1. *Частотой выполнения j-го линейного участка при заданных распределениях входных параметров называется величина* $e(b_j) = EY_j = \sum\limits_{(\hat{x}_1, \ldots, \hat{x}_p) \in M_p} \Pi_j(\hat{x}_1, \ldots, \hat{x}_p) \cdot P((X_1, \ldots, X_p) = (\hat{x}_1, \ldots, \hat{x}_p))$ *– это математическое ожидание числа выполнений j-го линейного участка.*

Эта постановка задачи взята из [1, 2]. Там она была сформулирована в таком наиболее общем виде впервые. Позднее с развитием техники профилировки программ появились частные случаи этой постановки [3, 4]. В тех работах под частотой выполнения понималась следующая величина.

Определение 2. *Частотой выполнения j-го линейного участка называется величина* $n(b_j) = \frac{\sum\limits_{(\hat{x}_1,\ldots,\hat{x}_p)\in M_p} \Pi_j(\hat{x}_1,\ldots,\hat{x}_p)}{|M_p|}$ *– это среднее число выполнений j-го линейного участка на всех допустимых наборах входных данных.*

Однако следует отметить, что при условии дискретного равномерного распределения входных параметров величины из определений 1 и 2 совпадают, так как в этом случае $P((X_1,\ldots,X_p)=(\hat{x}_1,\ldots,\hat{x}_p)) = \frac{1}{|M_p|}$.

В данной работе под термином *частота выполнения* будем понимать величину из определения 1. При этом подразумевая, что эта частота зависит от распределений входных параметров, заданных в постановке задачи.

Возможно несколько подходов к вычислению частоты выполнения линейного участка $e(b_j)$. В работах [1, 2] представлены методы построения функций распределения числа выполнений линейных участков программы по распределениям входных параметров в аналитическом виде. Фактически используются формулы свертки и при реализации требуются значительные вычислительные затраты.

В данной работе предлагается новый подход: *не вычислять* точное значение частоты выполнения, так как это сопряжено с большими вычислительными затратами, сравнимыми с прогонами программы на всех входных данных, а *оценивать* значение частоты выполнения с заранее заданной точностью. Будем предполагать, что распределения значений входных величин X_i заданы, например, в виде датчиков случайных чисел, то есть в аналитическом виде эти распределения нам не доступны. Для оценки частоты выполнения линейного участка, которая заключается в приближенном вычислении математического ожидания случайной величины $Y_j = \Pi_j(X_1,\ldots,X_p)$, будем использовать метод статистических испытаний (метод Монте-Карло) [5]. Идея этого метода заключается в проведении многократных опытов с исследуемой случайной величиной Y_j, и на основе полученных значений делается вывод о различных характеристиках этой случайной величины.

3. Предлагаемый метод оценки частоты выполнения линейных участков программы

Необходимо оценить значение $e(b_j) = EY_j = E\Pi_j(X_1,\ldots,X_p)$ частоты выполнения линейного участка с заранее заданной точностью ε, то есть найти N_j : $|e(b_j) - N_j| \le \varepsilon$. Указанное неравенство должно выполняться с наперед заданной достоверностью γ.

Проведем n испытаний случайной величины $Y_j = \Pi_j(X_1,\ldots,X_p)$. На практике это означает запуски программы $\Pi(x_1,\ldots,x_p)$ на значениях, сгенерированных по функциям распределения случайных величин для входных параметров [5, 6] (см. п. 4). При каждом испытании используется новый экземпляр программы $\Pi(x_1,\ldots,x_p)$. Входные наборы $(\hat{x}_1,\ldots,\hat{x}_p)$ выбираются случайным образом и независимо от наборов, использованных на прошлых итерациях.

Получим выборку из n значений $Y_j^1, Y_j^2, \ldots, Y_j^n$ – числа выполнений j-го линейного участка при запусках программы на элементах из M_p, отвечающих распределениям $X_1,\ldots,X_p$.

Определим величину $N_j = 1/n \cdot \sum_{i=1}^{n} Y_j^i$, которую будем называть *искомой* оценкой частоты выполнения j-го линейного участка. Это следует из следующих утверждений.

Утверждение 1. *Пусть исходная программа* $\Pi(x_1, \ldots, x_p)$ *не зацикливается и ограничена по размеру (размер – это число линейных участков в графе потока управления программы). Тогда для выборки* $Y_j^1, Y_j^2, \ldots, Y_j^n$ *значений случайной величины* $Y_j = \Pi_j(X_1, \ldots, X_p)$ $(j = \overline{1,m})$ *верно, что она состоит из независимых, одинаково распределенных величин с конечными моментами любых конечных порядков.*

Доказательство. Надо доказать, что выборка $Y_j^1, Y_j^2, \ldots, Y_j^n, \ldots$ состоит из (1) независимых (2) одинаково распределенных величин с (3) конечными моментами любых порядков. Справедливость первых двух пунктов очевидна. Остановимся на доказательстве третьего пункта.

Напомним, что Y_j – дискретная случайная величина, так как число выполнений линейного участка при одном испытании принадлежит натуральным числам с нулем $N \cup \{0\}$. Множество значений входных параметров конечно, и ни на одном из них программа $\Pi(x_1, \ldots, x_p)$ не зацикливается. Программа $\Pi(x_1, \ldots, x_p)$ ограничена по размеру. Вышесказанное означает, что для любого линейного участка существует максимальная частота выполнения: $\exists N_{max}^j : \forall i \; Y_j^i \leq N_{max}^j$. Тогда возможные значения случайной величины Y_j образуют конечную последовательность чисел $0,\ 1,\ 2, \ldots,\ N_{max}^j$, которые она принимает с некоторыми вероятностями $P(Y_j = 0),\ P(Y_j = 1),\ P(Y_j = 2), \ldots,\ P(Y_j = N_{max}^j)$. Поэтому ряды

$$EY_j = \sum_{i=1}^{N_{max}^j} k \cdot P(Y_j = k),\ EY_j{}^2 = \sum_{i=1}^{N_{max}^j} k^2 \cdot P(Y_j = k),\ EY_j{}^3 = \sum_{i=1}^{N_{max}^j} k^3 \cdot P(Y_j = k), \ldots$$

всегда конечны, так как это конечные суммы ограниченных величин. Следовательно, существуют конечные моменты любых конечных порядков, так как в силу линейности математического ожидания центральные моменты могут быть выражены через начальные.

□

Это утверждение имеет важное следствие.

Следствие 1. *Если дополнительно предположить, что случайная величина* Y_j *имеет ненулевую дисперсию, то к выборкам вида* $Y_j^1, Y_j^2, \ldots, Y_j^n, \ldots$ *различных характеристик программы применимы Закон больших чисел*[1] *и Центральная предельная теорема*[2].

Следующее утверждение определяет взаимосвязь N_j и $e(b_j)$.

[1] **Закон больших чисел**. Пусть $X_1, X_2, \ldots$ – независимые одинаково распределенные случайные величины и $EX_1 = a$. Тогда с вероятностью единица выполняется соотношение $\lim\limits_{n\to\infty} 1/n \sum\limits_{k=1}^{n} X_k = a$ [6].

[2] **Центральная предельная теорема**. Пусть $X_1, X_2, \ldots$ – независимые одинаково распределенные случайные величины и $EX_1 = a$, и $DX_1 = \sigma^2$, причем $0 < \sigma^2 < \infty$. Тогда при $n \to \infty$ наибольшее по x значение величины $|P(\frac{\sum_{k=1}^{n} X_k - na}{\sigma\sqrt{n}} < x) - \Phi(x)|$ стремится к нулю [6].

Утверждение 2. *Если для программы* $\Pi(x_1, \ldots, x_p)$ *выполнены условия утверждения 1 и следствия 1, то справедливо следующее:*

1. $N_j \to e(b_j)$ *при* $n \to \infty$.

2. *При достаточно больших* n *с достоверностью, близкой к* γ, *можно утверждать, что* $|e(b_j) - N_j| \le \frac{\sigma_j}{\sqrt{n}} \cdot u_{\frac{1+\gamma}{2}}$,
где $\sigma_j^2 = DY_j > 0$ – *дисперсия случайной величины* Y_j, $u_{\frac{1+\gamma}{2}}$ – *квантиль порядка* $\frac{1+\gamma}{2}$ *стандартного нормального закона.*

Доказательство. Из следствия к утверждению 1 вытекает, что к ряду из чисел выполнения $Y_j^1, Y_j^2, \ldots, Y_j^n, \ldots$, полученных после испытаний случайной величины $Y_j = \Pi_j(X_1, \ldots, X_p)$, можно применять *Закон больших чисел* и *Центральную предельную теорему*, из которых явным образом следует верность первого и второго пунктов утверждения соответственно. □

Из первого пункта утверждения 2 следует, что при увеличении числа прогонов программы значение оценки частоты выполнения линейного участка N_j будет приближаться к частоте $e(b_j)$.

Второй пункт утверждения 2 позволяет оценить число прогонов программы, необходимое для вычисления $e(b_j)$ с заданной точностью и достоверностью. Если в какой-то момент проведения испытаний выполнено условие $n > \left(\frac{u_{\frac{1+\gamma}{2}}}{\varepsilon}\right)^2 \cdot \sigma_j^2$, то это означает, что частота выполнения была вычислена с заранее заданной точностью ε и достоверностью γ, то есть удалось найти такую величину N_j, что $P(|e(b_j) - N_j| \le \varepsilon) = \gamma$.

Заметим, что из Центральной предельной теоремы вытекает, что при больших n распределение суммы случайных величин можно аппроксимировать нормальным законом. На практике необходимо проверять адекватность такой нормальной аппроксимации. Это можно сделать с помощью теоремы Берри–Эссеена[3] [7] (теорема применима к данной задаче – см. утверждение 1 и следствие 1). Из теоремы следует, что на практике для применимости нормальной аппроксимации необходимо, чтобы $\frac{C_0 \mu_j^3}{\sigma_j^3 \sqrt{n}} \le \varepsilon'$, где ε' – задаваемая точность нормальной аппроксимации, которую можно выбирать из условия $\varepsilon' < (1-\gamma)c$, $0 < c < 1$. Например, $\varepsilon' = \frac{1-\gamma}{10}$.

На практике σ_j^2 и μ_j^3 не известны. В этом случае при достаточно большом числе испытаний ($n \ge 30$) вместо них можно использовать оценки $s_j^2 = \frac{1}{n-1} \cdot \sum\limits_{i=1}^{n} (Y_j^i - N_j)^2$ (выборочная дисперсия) и $m_j^3 = \frac{1}{n-1} \cdot \sum\limits_{i=1}^{n} (Y_j^i - N_j)^3$ (выборочный третий центральный момент) соответственно [5]. Конкретное значение достаточного числа испытаний зависит от задачи. В данной работе в качестве порога берется 30 испытаний.

[3] **Теорема Берри–Эссеена**. Предположим, что выполнены условия Центральной предельной теоремы для независимых одинаково распределенных случайных величин и существует $\mu^3 = E|X_1 - a|^3$, то $\sup\limits_x |P(\frac{\sum_{k=1}^n X_k - na}{\sigma\sqrt{n}} < x) - \Phi(x)| \le \frac{C_0 \mu^3}{\sigma^3 \sqrt{n}}$, где C_0 – абсолютная положительная постоянная, $0.4097... < C_0 \le 0.4784$ [7].

Сформулируем всё вышесказанное подробнее в следующем алгоритме.
Алгоритм 1.

1. Задать ε, γ.
2. Инициализировать счетчик числа испытаний нулем, $n = 0$.
3. Провести одно испытание случайной величины Y_j с результатом Y_j^n и увеличить счетчик числа испытаний на единицу, $n = n + 1$.
4. Если $n > 30$, то вычислить текущие значения оценки частоты выполнения j-го линейного участка N_j, выборочной дисперсии s_j^2 и выборочного третьего момента β_3. Иначе, провести еще один прогон программы (пункт 2).

   (a) $N_j = 1/n \cdot \sum\limits_{i=1}^{n} Y_j^i$,

   (b) $s_j^2 = \frac{1}{n-1} \cdot \sum\limits_{i=1}^{n} (Y_j^i - N_j)^2$,

   (c) $m_j^3 = \frac{1}{n-1} \cdot \sum\limits_{i=1}^{n} (Y_j^i - N_j)^3$.

5. Если $n > \left(\frac{u_{\frac{1+\gamma}{2}}}{\varepsilon} \right)^2 \cdot s_j^2$ и $\frac{C_0 m_j^3}{s_j^3 \sqrt{n}} \leq \frac{1-\gamma}{10}$, то N_j является искомой оценкой $e(b_j)$ с заданной точностью ε и достоверностью γ. Иначе, провести очередной прогон программы (пункт 2).

Отметим, что данный алгоритм применим не всегда. А именно, на практике возможны ситуации, когда распределение входных данных таково, что рассматриваемый линейный участок программы практически не активизируется при прогонах, что приводит либо к статистическому выводу о нарушении условий центральной предельной теоремы, либо к статистическому выводу о неадекватности нормальной аппроксимации. Рассмотрим пример того, как в подобных случаях на практике принимается решение о применимости (или неприменимости) описанного выше алгоритма.

Рассмотрим программу, тело которой состоит из одного условного оператора, где есть одна входная переменная x типа *long integer*, занимающая 64 бита. Распределение значений входной переменной – дискретное равномерное, то есть $\forall \hat{x} \in T(x) \;\; P(x = \hat{x}) = \frac{1}{2^{64}}$.

$$\boxed{if\ (x\ ==\ a)\ x\ =\ 0;\ else\ x\ =\ 1;}$$

В этом примере с вероятностью $1 - \frac{1}{2^{64}}$ (то есть практически всегда) будет выполняться *else*-ветка. Особо следует заметить, что формально предполагается, что распределение входных данных не известно программисту-исследователю, поэтому вывод о применимости или неприменимости алгоритма, описанного выше, он делает *только* на основании статистических данных. При этом возможны следующие варианты.

1. После n прогонов счетчик числа выполнений *else*-ветки равен n. В этом случае выборочная дисперсия равна нулю, и следует сделать статистический вывод о том, что условия Центральной предельной теоремы не выполнены. При этом практическая достоверность такого вывода равна $\left(1 - \frac{1}{2^{64}}\right)^n$, что при n порядка десятков практически равно единице.

2. После n прогонов программы счетчик числа выполнений *else*-ветки равен $n-1$. В этом случае для *then*-ветки выборочное среднее равно $\frac{1}{n}$, выборочная дисперсия равна $\frac{1}{n} - \frac{1}{n^2}$, то есть следует сделать статистический вывод о том, что формально условия Центральной предельной теоремы выполнены. Однако при этом третий абсолютный выборочный момент оказывается равным $(n^3 - 3n^2 + 4n - 2)/n^4 = O(\frac{1}{n})$, что приводит к тому, что правая часть неравенства Берри–Эссеена оказывается равной примерно 0.5 при любых n, то есть необходимо сделать практический вывод о неадекватности нормальной аппроксимации.

3. После n прогонов программы счетчик числа выполнений *else*-ветки равен $n-2$ или меньше. Такой вариант в рассматриваемых условиях имеет пренебрежимо малую вероятность и потому практически невозможен. Но если все же такой вариант реализовался, то следует поступать так же, как описано в п. 2.

Таким образом, при исследовании указанной программы с описанным выше распределением входных данных с достоверностью, практически равной единице, будет сделан вывод о том, что к ней не применим алгоритм 1.

На практике в любой программе будет хотя бы один линейный участок, частота выполнения которого не будет зависеть от значений исходных данных. Например, блок инициализации программы, то есть стартовый блок, где инициализируются переменные программы. Поэтому в тех случаях, когда частота линейного участка оказалась постоянной на протяжении заданного числа испытаний программы, окончательное решение о частоте выполнения таких блоков должно оставаться за программистом.

4. Практическое применение метода

Описанный метод был реализован в экспериментальной системе *FreqSys*. Входными данными *FreqSys* служат текст программы на языке Си, функции распределения значений входных параметров программы, диапазоны их значений, точность вычисления и достоверность оценки частоты выполнения линейных участков. В качестве результата система выдает список линейных участков программы с указанием оценки их частоты выполнения.

FreqSys написана на языке C++ как проход в компиляторе *LLVM* [8]. Работа программы состоит из двух этапов: изменение исследуемой программы и прогоны измененной программы.

Исходная программа преобразуется следующим образом (Рисунок 1):

1. В программу добавляются функции инициализации, которые присваивают входным параметрам программы значения, сгенерированные по закону распределения их значений [5, 6].

2. Команды чтения из внешних источников (с клавиатуры и т.п.) входных параметров программы заменяются на обращение к функциям инициализации.

3. В начало каждого линейного участка добавлены счетчики. Счетчики увеличиваются на единицу каждый раз, когда происходит переход управления к линейным участкам. После одного прогона программы счетчики будут содержать количество передач управления на каждый линейный участок.

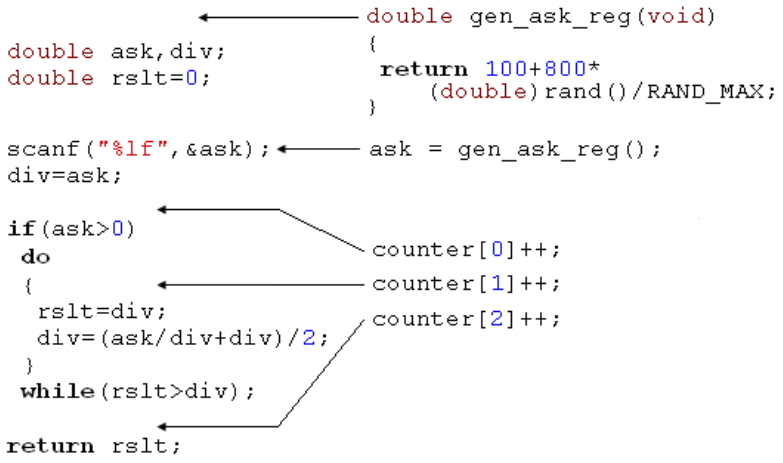
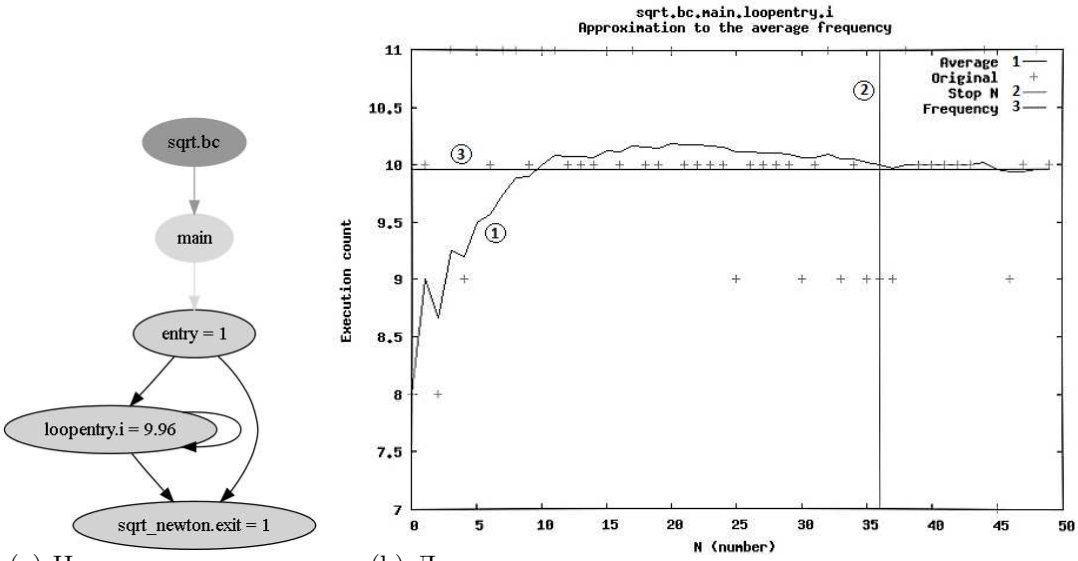


Рис. 1. Преобразования исходной программы

Не трудно видеть, что эти преобразования не затрагивают функциональность исходной программы, хотя несколько изменяют её временные характеристики.

Прогоны измененной программы производятся в соответствии с описанным выше алгоритмом (см. п. 3). Заметим, что результаты каждого прогона будут отличаться от результатов предыдущего, поскольку каждый раз при работе программы происходит генерация новых значений входных параметров.

Рассмотрим работу *FreqSys* на примере программы вычисления квадратного корня итерационным методом Ньютона (Рисунок 1). В качестве распределения единственного входного параметра ask используется равномерное распределение на отрезке $[100, 800]$, то есть $ask \sim R(100, 800)$. Необходимо оценить частоты линейных участков программы с точностью $\varepsilon = 0.3$ и достоверностью $\gamma = 0.95$. После работы *FreqSys* выдает граф потока управления исследуемой программы с указанием вычисленных частот выполнения каждого линейного участка (Рисунок 2a). В данном примере частота выполнения тела цикла равна 9.96 итерациям. Для каждого линейного участка можно посмотреть динамику вычисления оценки его частоты (Рисунок 2b). Видно, что частота выполнения тела цикла вычислена с заданной точностью уже на 36-й итерации.



(a) Частоты выполнения (b) Динамика вычисления оценки частоты выполнения

Рис. 2. Демонстрация работы *FreqSys*

Укажем достоинства *FreqSys* по сравнению со стандартными профилировщиками компиляторов *LLVM* [8], *GCC* [9]. Профилировщики занимаются сбором информации об одном прогоне программы. Пользователь запускает исследуемую программу, вводит некоторые значения входных параметров и на выходе получает информацию о том, сколько раз управление переходило на тот или иной участок программы. Сразу отметим, что, используя указанные профилировщики, можно вычислять частоты выполнения линейных участков в соответствии с описанным выше подходом, но для пользователя это является трудоемкой задачей.

1. Для вычисления частот выполнения линейных участков программы при использовании профилировщиков пользователю необходимо n ($n \to \infty$) раз запустить программу, ввести необходимые входные данные, получить результат. При использовании *FreqSys* пользователю необходимо лишь один раз задать распределения значений входных данных программы.

2. *FreqSys* позволяет исследовать программы с неразрешенными зависимостями по внешним переменным. Для внешних переменных необходимо только задать распределения значений. При использовании же стандартных профилировщиков программа должна быть исполняемой.

3. *FreqSys* позволяет исследовать отдельные функции без прогона всей программы. Для этого надо задать распределения значений для формальных аргументов анализируемой функции и используемых данной функцией глобальных переменных. При использовании же стандартных профилировщиков нельзя проанализировать отдельную функцию без запуска всей программы.

5. Заключение

В данной работе описан метод оценки частоты выполнения линейного участка программы, который в отличие от существующих позволяет по заданной точности оценки частоты определить количество испытаний программы на разных входных данных.

Определена область его адекватного применения. Программа должна быть конечна и не зацикливаться на всех допустимых входных данных. Метод не должен применяться к линейным участкам, частоты выполнения которых не зависят от входных данных, и вероятность их срабатывания близка к нулю. Поэтому линейные участки с постоянной частотой выполнения требуют отдельного рассмотрения.

Описанный подход реализован в экспериментальной системе *FreqSys*, которая применяется в разрабатываемой автором системе компактного представления программ для встроенных систем и показывает хорошие результаты работы [10].

Список литературы

1. *Smelianski R.L., Alanko T.* On the calculation of control transition probabilities in a program Inform // Processing Letters. 1986. N.3.

2. *Смелянский Р.Л., Гурьев Д.Е., Бахмуров А.Г.* Об одной математической модели для расчета динамических характеристик программы // Программирование. 1986. N6.

3. *Youfeng Wu, James R. Larus* Static Branch Frequency and Program Profile Analysis // Proceedings of the 27th annual international symposium on Microarchitecture. 1994. P. 1–11.

4. *Thomas Ball , James R. Larus* Optimally profiling and tracing programs // ACM Transactions on Programming Languages and Systems (TOPLAS). 1994. P. 1319–1360.

5. *Скрипкин В.А., Моисеенко Е.А.* Математические методы исследования операций в военном деле. М.: Военное издательство министерства обороны СССР, 1979.

6. *Гнеденко Б.В.* Курс теории вероятностей. 7-е изд. М.: УРСС, 2001. 448 с.

7. *Korolev V., Shevtsova I.* An improvement of the Berry-Esseen inequality with applications to Poisson and Mixed Poisson random sums // Scandinavian Actuarial Journal. 2010 (to appear).

8. Documentation for the LLVM System [HTML] (http://llvm.cs.uiuc.edu/docs/).

9. The GNU Compiler Collection [HTML] (http://gcc.gnu.org/).

10. *Шалимов А.В.* Метод компактного представления программ на основе частотных характеристик их поведения // Интеллектуализация обработки информации: Тезисы докладов Международной научной конференции / Крымский научный центр НАН Украины. Симферополь, 2008.

A method of determining the execution frequency of program basic blocks

Shalimov Alexander

Keywords: execution frequency, program analysis, Monte Carlo method, profiling.

The goal of this article is to consider the task of determining the execution frequency of program basic blocks. This task is important for such applications as program optimization, paralleling program execution, computing resources allocation, program compaction, and malicious software detection. A new method is proposed in the article for evaluation of basic block execution frequency based on the Monte Carlo method. The method proposed allows us to estimate a number of program runs to get the execution frequency with a given precision.

Сведения об авторе:
Шалимов Александр Владиславович,
МГУ им. М. В. Ломоносова,
аспирант факультета вычислительной математики и кибернетики

От редакции. Статья рекомендована к публикации рецензентом — профессором факультета вычислительной математики и кибернетики МГУ, доктором физико-математических наук В. Ю. Королёвым.