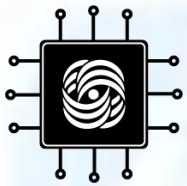
Анализ поведения программ

Лаб.Вычислительных комплексов,
каф.АСВК, ф-та ВМиК МГУ

Проф.Смелянский Р.Л.

Содержание

- Природа программ
- О различных толкованиях поведения программы
  - Оценка производительности
  - Синтез структуры вычислителя
  - WCET
  - Верификация программ
  - Отладка и тестирование программ
  - Оптимизация, декомпиляция, планирование



О частотных характеристиках выполнения кода последовательной программы

Введение

- Задача определения частоты выполнения линейных участков программы
- Ядро программы
- Приложения
  - оптимизация, распараллеливание программ,
  - выбор резидентной части в системах реального времени,
  - планирование нагрузки в многопроцессорных и многоядерных вычислителях,
  - выбор надлежащих эвристик в алгоритмах look ahead при выборе ветки в условных операторах.

Основные понятия и определения

- $\Pi(v_1,...,v_{In}) = (A_\Pi, E_\Pi)$ – *исходная программа с входными переменными* $v_1,...,v_{In}$,
  где $A_\Pi = \{b_j\}_{j=1}^k$ – *линейные участки,* $E_\Pi = \{(b_i, b_j)\}$ – *передачи управления*

- $V_\Pi = V_{In} \cup V_{Out} \cup V_{Iner} = \{v_i\}_{i=1}^m$ – *множество всех переменных программы* Π

- $T(v_i)$ – *тип переменной* $v_i \in V_\Pi$, *т.е. множество значений переменной* v_i
  $T_\Pi = T(v_1) \times ... \times T(v_m)$ – *множество всех состояний памяти программы* Π
  T_{In} – *декартово произведение* $T(v)$ *для всех* $v_i \in V_{In}$

- $\#\Pi_j(< V_{In} >)$ – *число выполнений блока* $b_j \in A_\Pi$ *на наборе значений* $< V_{In} >$

Определим n_{b_j} – *частоту выполнения* $b_j \in A_\Pi$, *как*

$$n_{b_j} = \frac{\#\Pi_j}{|T_{In}|}, \text{ где } \#\Pi_j = \sum_{\{<V_{In}> \in T_{In}\}} \#\Pi_j(< V_{In} >)$$

– *среднее число выполнения* b_j, *где усреднение взято по множеству* T_{In}.

Основные понятия и определения

- V_{In} – *случайная величина*, $< V_{In} >$ – *значение с.в.* V_{In}

  $\#\Pi_j(< V_{In} >)$ – *функция от с.в.* V_{In} *на* T_{In}

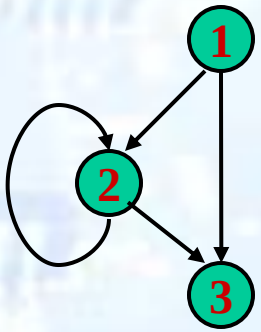
- *В терминах случайных величин частоту выполнения можно определить как*

$$E_{b_j} = E\#\Pi_j(< V_{In} >) = \sum_{< V_{In} >_k \in M_\Pi} \#\Pi_j(< V_{In} >) \cdot p(V_{In} =< V_{In} >_k),$$

где $p(V_{In} =< V_{In} >_k)$ – *вероятность, что* V_{In} *примет значение* $< V_{In} >_k$

Графовая модель [1]

```
y0 = 0;  y = x;
while  abs(y - y0) > eps  do
   begin y0 = y; y = y0 – x / y0  end
```



- **Постановка задачи**
  - Для каждой входной переменной известна функция распределения её значений.

$$D_v(x) = p(v = x) \qquad v \in V_{In}$$

  - Значения входных переменных статистически независимы
  - Требуется вычислить E_{b_j} для всех линейных участков.

В работе[1] *показано, как по* $D_v(x)$ *для* $v \in V_{In}$ *можно в аналитическом виде получить вид* $D_v(x)$ *для* $v \in \{V_{Iner} \cup V_{Out}\}$, *а зная эти функции получить вероятности переходов на дугах графа управления*.

- **Условия**
  - Тип всех переменных перечислим.
  - Значения любой переменной – есть рациональная функция от входных переменных, задаваемая в виде арифметического выражения, в котором допускаются только элементарные функции.

1. R.L. Smelianski, T. Alanko. *On the calculation of control transition probabilities in a program.* Inform. Processing Letters N.3, 1984

Математическая модель [2]

- В работе [2] была построена математическая модель, позволившая по статическому представлению *П* получить характеристики ее поведения.

$$(T, S, \Pi) - \textit{представление программы}$$

$$T - \textit{значения переменных}$$

$$S = \{s_j\}_{j=1}^{m} - \textit{операторы программы}$$

$$\Pi = (\pi_{ij}) - \textit{матрица предикатов переходов } i, j \in [1, m]$$

$$D^{(n)}(x) = (d_1^{(n)}(x), \ldots, d_m^{(n)}(x)) - \textit{распределение состояния процесса},$$

$$\textit{где } d_j^{(n)}(x) - \textit{вероятность работы } s_j \textit{ над } x \in T \textit{ на шаге } n.$$

$$D^{(n)}(x) \to D^{(n+1)}(x) - \textit{процесс функционирования модели}.$$

- Как показали эксперименты, выполненные еще в 80-е годы, решения рассматриваемой задачи, сформулированные в [1,2], обладают большой вычислительной сложностью и трудны для применения на практике.

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

Профилировка

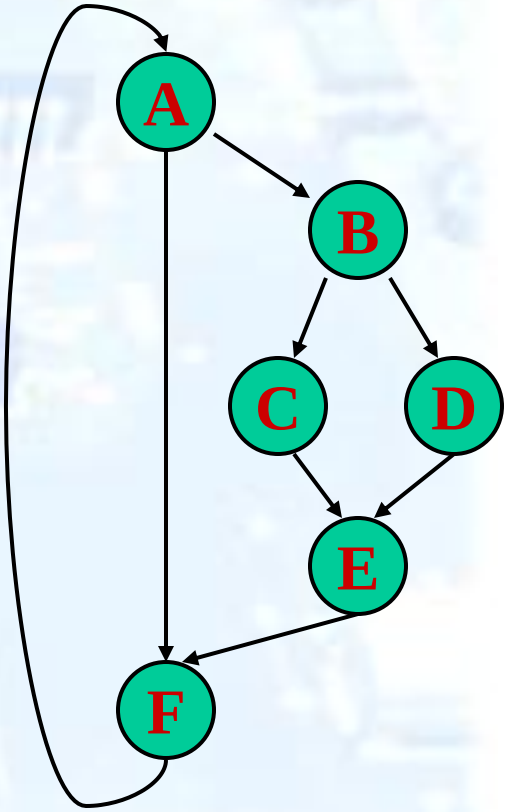
Профилировка программ – *это сбор разнообразной информации об исполненных командах, частоте их встречаемости, ветвлениях в операторах управления и частоте используемых веток, используемых переменных и областях памяти, хранимым там значениям.*

Профилировку можно разделить на следующие виды:

- по *типу собираемой информации*:
  - команд
  - данных
- по *способам получения информации*:
  - динамическую
  - статическую

Динамическая профилировка

- **Простой подход**
  - размещении счетчиков в каждом линейном участке программы,
  - прогон программы на различных входных значениях.
- Недостатки
  - много счетчиков,
  - много операций со счетчиками.
- Оптимальная расстановка счетчиков
  - уменьшения количества счетчиков за счет использования различных *зависимостей по управлению* в программе (число счетчиков сокращается в **2 раза**),
  - уменьшение числа операций со счетчиками за счет расстановки счетчиков в линейные участки *на менее вероятных переходах* (число операций сокращается в **4 раза**).



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

Статическая профилировка

Основная идея:

- Для каждого ветвления определяют эвристики для оценки вероятности переходов в программе
  - сработает переход *на новую итерацию цикла - 88%,*
  - сравнение *указателя с NULL или сравнение двух указателей* будет неуспешно - *60%,*
  - сравнение *переменной целого типа с нулем или на равенство константе* будет неуспешно - *84%,*
  - сработает переход на линейный участок*, содержащий команду return - 28%,*
  - сработает переход на линейный участок, *содержащий обработку исключительной ситуации* - 22%.
- Вычисляют итоговые вероятности переходов по вероятностям эвристик (*теория Демпстера-Шафера*)
- Определяют частоты выполнения линейных участков программы

Статический профиль, построенный с помощью описанного подхода, близок к реальному профилю программы **с *20%* точностью**.

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

Метод оценки частоты выполнения линейных участков программы

Необходимо оценить значение E_{b_j} *с заранее заданной точностью* ε, *т.е. найти*

$$n'_{b_j} : \left| E_{b_j} - n'_{b_j} \right| < \varepsilon$$

- **Предлагаемый подход** основан на методе статистических испытаний (метод Монте-Карло), состоит в многократных испытаниях случайной величины $\#\Pi_j(<V_{In}>)$ с целью оценки E_{b_j}

Проведем k *испытаний программы* Π *на множестве* M_{Π} *случайной величины* V_{In}. *Получим выборку из* k *значений* $\#\Pi_j(<V_{In}>_i)$.

- Введем величину

$$n'_{b_j} = \frac{1}{k}\sum_{i=1}^{k} \#\Pi_j(<V_{In}>_i)$$

Метод оценки частоты выполнения линейных участков программы (2)

- **Теорема 1**

Пусть дана программа П, которая:

1. *Не зацикливается, т.е. длина вычислительного процесса конечна,*
2. $|A_П|$ *– конечно,*
3. *Все* $v \in V_{In}$ *статистически независимы.*

Тогда для выборки $\{\#П_j(<V_{In}>)_i\}_{i=1}^{k}$ *верно, что величины* $\#П_j(<V_{In}>)_i$

- *статистически независимы,*
- *одинаково распределены,*
- *имеют конечное математическое ожидание и дисперсию.*

- **Следствие**

К выборкам вида $\{\#П_j(<V_{In}>)_i\}_{i=1}^{k}$ *применим Закон Больших Чисел и Центральная Предельная Теорема.*

Метод оценки частоты выполнения линейных участков программы (3)

- **Теорема 2**

  Если для программы П выполнены условия теоремы 1, *то справедливы следующие утверждения*:

  - $n'_{b_j} \to E_{b_j}$ *при* $k \to \infty$,
  - *С достоверностью* λ *можно сказать*

$$\left| E_{b_j} - n'_{b_j} \right| \le \frac{s_j}{\sqrt{k}} \cdot u_{\frac{1+\lambda}{2}},$$

$$\text{где } s_j^2 = \frac{1}{k-1} \cdot \sum_{i=1}^{k} (E_{b_j} - n'_{b_j})^2,\ u_{\frac{1+\lambda}{2}} - \text{квантиль порядка } \frac{1+\lambda}{2}.$$

- **Достаточно ли числа прогонов программы?**

$$\text{Если } k > \left(\frac{u_{\frac{1+k}{2}}}{\varepsilon} \right)^2 \cdot s_j^2, \quad \text{то } n'_{b_j} \text{ является искомой оценкой } E_{b_j}.$$

Алгоритм определения частот выполнения линейных участков

1. $N = 0$ – *счетчик числа прогонов*;

2. *Проводим один прогон программы*: *для всех* $b_j \to n_j^N$. $N = N + 1$;

3. *Если* $N > 1$, *то вычислить* n'_{b_j} *и* s_j^2 *для всех рассматриваемых* b_j.

*Иначе п.*2.

$$n'_{b_j} = \frac{1}{N} \cdot \sum_{i=1}^{N} n_j^i,$$

$$s_j^2 = \frac{1}{N-1} \cdot \sum_{i=1}^{N} (n_j^i - n'_{b_j})^2.$$

4. *Для каждого рассматриваемого* b_j *проверяется* $N > \left(\frac{u_{\frac{1+\lambda}{2}}}{\varepsilon} \right)^2 \cdot s_j^2$:

если условие выполнено, *то* n'_{b_j} – *искомая оценка и* b_j *убирается из рассматриваемых линейных участков.*

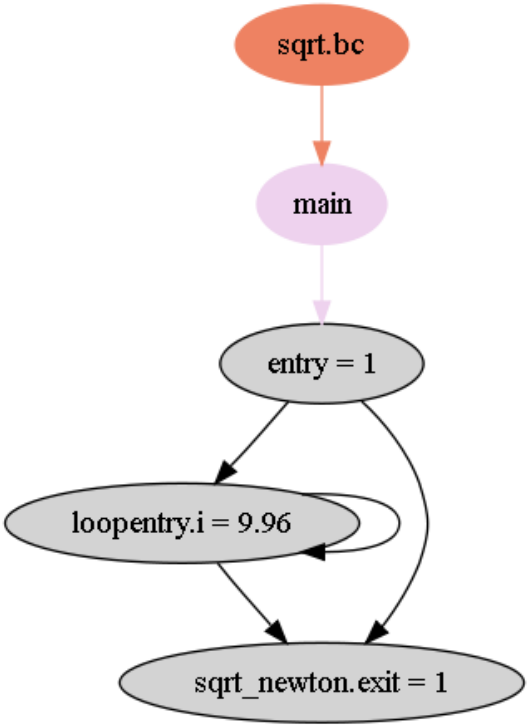
*Если остались рассматриваемые линейные участки, то п.*2.

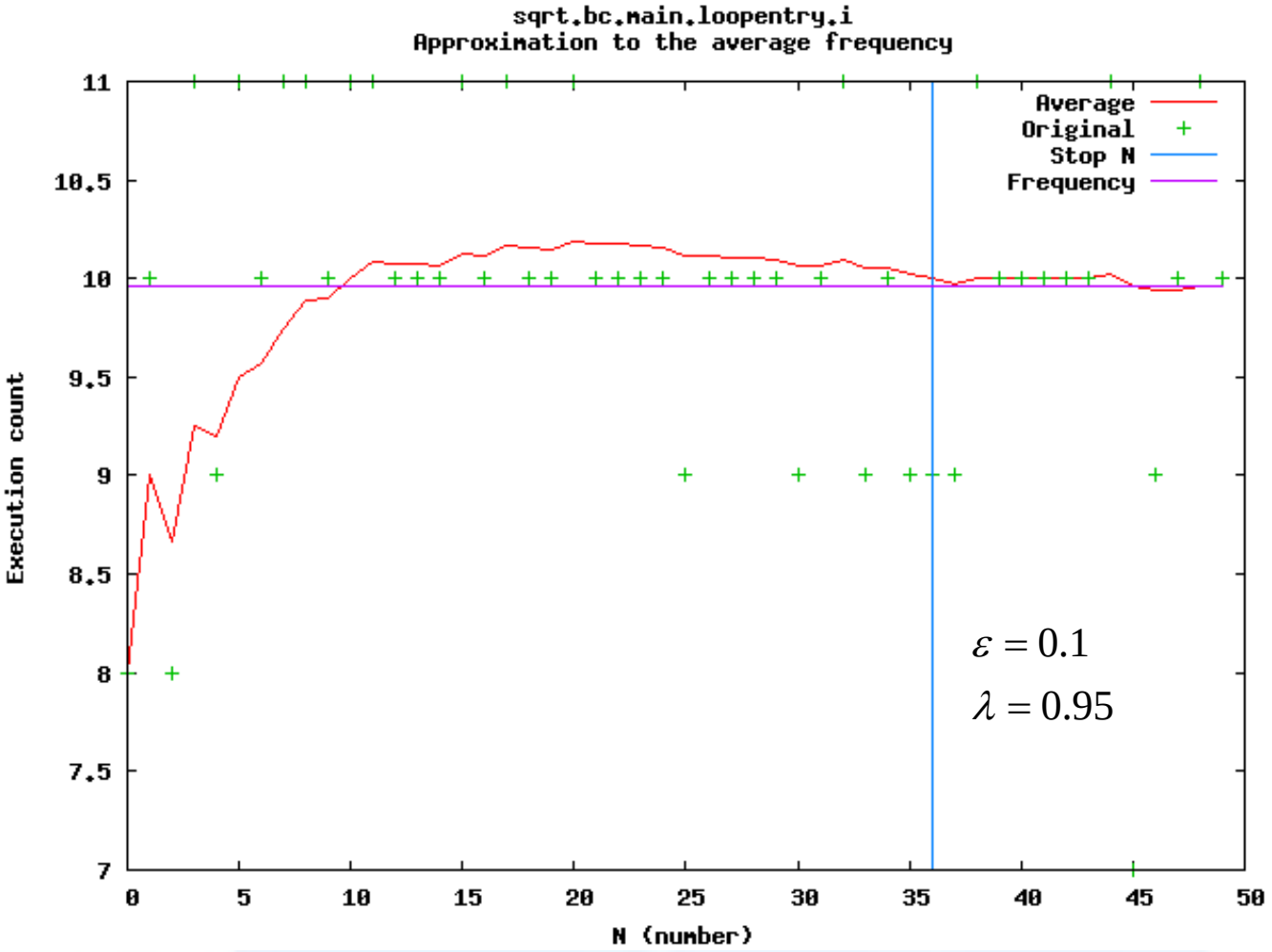
Реализация метода оценки частот выполнения линейных участков программы (FreqSys)

- Вычисление квадратного корня итерационным методом Ньютона

```
double gen_ask_reg(void)
{
 return 100+800*
     (double)rand()/RAND_MAX;
}

double ask,div;
double rslt=0;

scanf("%lf",&ask);
div=ask;

ask = gen_ask_reg();

if(ask>0)
 do
 {
  rslt=div;
  div=(ask/div+div)/2;
 }
 while(rslt>div);

return rslt;

counter[0]++;
counter[1]++;
counter[2]++;
```

Пример работы FreqSys





Результаты экспериментов

SNU RealTime	**Описание**	LOC	$\lvert A_П \rvert$	$\lvert E_П \rvert$	$\lvert T_{In} \rvert$	M	ε^*
minver	*Построение обратной матрицы (размер 10x10, значения 0-100)*	200	41	60	10^{200}	**70**	0.04
qsort	*Быстрая сортировка (размер 20, значения 0-100)*	150	30	50	20^{100}	**80**	0.08
select	*Поиск 10-го наибольшего элемента массива (размер 20, значения 0-100)*	100	23	40	20^{100}	**100**	0.06
prime	*Определение простоты числа (значения 0-1000000)*	50	6	10	10^6	**90**	0.03
sqrt	*Вычисление квадратного корня итерационным методом (значения 0-1000)*	30	3	4	10^3	**30**	0.02

- *Желаемая точность оценки* $\varepsilon = 0.3.$
- *Достоверность оценки* $\lambda = 0.95.$
- *Равномерное распределение входных параметров.*

ε^* – *отклонение оценок на* M *прогонах и на* 1000 *прогонах*

Заключение

- Дана классификация методов оценки частотных характеристик поведения последовательных программ
- Проведен сравнительный анализ этих методов
- Разработан новый подход к оценке частотных характеристик поведения последовательных программ, основанный на методе Монте-Карло, и позволяющий определять число испытаний, исходя из требуемой точности оценки.
- Сформулирован список открытых вопросов

Список открытых вопросов

- Как определять функцию распределения для производных типов данных?
- Влияние предположения о статистической независимости входных параметров на оценку характеристик поведения программы
- Методика выбора набора исходных данных, если неизвестно распределение входных параметров?
- Какая будет погрешность оценки, если предположение о распределении входных значений будет не верным?
- Распространение созданной техники на случай распределенных программ.

Спасибо за внимание

e-mail: smel@cs.msu.su

Достоинства FreqSys

- Стандартные профилировщики: gprof, llvm-prof
- Преимущества FreqSys
  - Автоматизация получения оценки частоты выполнения линейных участков
  - Возможность исследования программ с неразрешенными зависимостями по внешним переменным
  - Возможность исследования отдельных функций без прогонов всей программы