Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Дискретный анализ»

Студент: Д. Д. Наумов
Преподаватель: А. А. Кухтичев
Группа: М8О-306Б

Дата:
Оценка:
Подпись:

Москва, 2019

Лабораторная работа №3

Задание: Для реализации словаря из предыдущей лабораторной работы необходимо провести исследование скорости выполнения и потребления оперативной памяти. В случае выявления ошибок или явных недочётов, требуется их исправить.

Результатом лабораторной работы является отчёт, состоящий из:

Дневника выполнения работы, в котором отражено что и когда делалось, какие средства использовались и какие результаты были достигнуты на каждом шаге выполнения лабораторной работы.

Выводов о найденных недочётов.

Сравнение работы исправленной программы с предыдущей версии.

Общих выводов о выполнении лабораторной работы, полученном опыте.

Минимальный набор используемых средст должен содержать утилиту gprof и библиотеку dmalloc, однако их можно заменять на любые другие аналогичные или более известные утилиты (например, Valgrind или Shark) или добавлять к ним новые (например, gcov).

Вариант: PATRICIA

1 Valgrind

Тест состоит из ста тысяч случайных команд.

```
dima@dima-System-Product-Name:~/Study/DA/lab2/solution: valgrind --leak-check=full
--leak-resolution=med ./solution <../test/tests/01.t >/dev/null
==25374== Memcheck,a memory error detector
==25374== Copyright (C) 2002-2017,and GNU GPL'd,by Julian Seward et al.
==25374== Using Valgrind-3.14.0 and LibVEX; rerun with -h for copyright info
==25374== Command: ./solution
==25374==
==25374==
==25374== HEAP SUMMARY:
==25374==     in use at exit: 122,880 bytes in 6 blocks
==25374==   total heap usage: 865,161 allocs,865,155 frees,2,792,261 bytes
allocated
==25374==
==25374== LEAK SUMMARY:
==25374==    definitely lost: 0 bytes in 0 blocks
==25374==    indirectly lost: 0 bytes in 0 blocks
==25374==      possibly lost: 0 bytes in 0 blocks
==25374==    still reachable: 122,880 bytes in 6 blocks
==25374==         suppressed: 0 bytes in 0 blocks
==25374== Reachable blocks (those to which a pointer was found) are not shown.
==25374== To see them,rerun with: --leak-check=full --show-leak-kinds=all
==25374==
==25374== For counts of detected and suppressed errors,rerun with: -v
==25374== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Как видно утечек памяти нет.

2 Gprof

Тест состоит из миллиона вставок. Для удобства чтения была удалена одна колонка(total)

```
Flat profile:

Each sample counts as 0.01 seconds.
%   cumulative   self              self
time   seconds   seconds    calls  ms/call   name
29.03      0.74     0.74 207602041    0.00   PatriciaTreeNode::get_bit_index()
23.54      1.34     0.60 131154664    0.00   string::digit(int)
7.85      1.54     0.20  1000000     0.00   PatriciaTree::insert(string&,unsigned
long long)
6.28      1.70     0.16  1000000     0.00   char* std::transform<char*,char*,int
(*)(int)>(char*,char*,char*,int (*)(int))
5.88      1.85     0.15 30512025     0.00   PatriciaTreeNode::get_right()
3.14      1.93     0.08 34710838     0.00   string::at(int) const
2.75      2.00     0.07 36893347     0.00   PatriciaTreeNode::get_key()
```

Как видно, чаще всего вызывается операция взятия бита, т.к. она позволяет избавиться от лишних сравнений ключа.

3 Выводы

Профилирование и инструментирование очень сильно помогает при обнаружении ошибок в программе и ее оптимизации. Например уже с первой лабораторной работы я использовал профилирование и после анализа изменил оператор равно в строке и она стала работать быстрее. Очень удобной и часто применяемой является утилита valgrind, с помощью неё можно очень быстро понять есть ли утечка в программе и где она возникает, а также такие ошибки как неинициализированные переменные, невалидные запись и чтение.