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. А. Тарпанов
Преподаватель: Н. С. Капралов
Группа: М8О-204Б
Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа №3

Задача: 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.
Минимальный набор используемых средств 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).

1 Benchmark

Для тестирования производительности программы напишем benchmark, в котором будем сравнить скорость работы std::map и моей реализации AVL-дерева. Программа выводит суммарное затраченное время для каждой операции. Тест содержит в себе один миллион команд.

```
kng@Legion:/mnt/c/vsc/da/lab2/bench$ ./benchmark <test1.txt
==============START============
INSERT std::map time: 141 ms
INSERT AVL-tree time: 636 ms
==============================
DELETE std::map time: 221 ms
DELETE AVL-tree time: 733 ms
==============================
SEARCH std::map time: 117 ms
SEARCH AVL-tree time: 250 ms
==============================
```

Анализируя результаты, видно, что моё AVL-дерево проигрывает в скорости во всех операциях, однако разрыв не настолько большой, как можно было ожидать. Наибольшая разница во времени наблюдается в операции вставки. Вставка в AVL-дерево в 4.5 раза медленнее, чем вставка в std::map. Вероятно, что с ростом количества команд разрыв между этими контейнерами будет увеличиваться.

2 Valgrind

Valgrind - программа для поиска ошибок обращения с памятью, поиска утечек памяти и профилирования. С помощью Valgrind можно оценить количество памяти, используемой программой. Запустим Valgrind на тесте с миллионом команд.

```
kng@Legion:/mnt/c/vsc/da/lab2/solutionn$ valgrind ./solution <test1.txt >dump.txt
==706== Memcheck,a memory error detector
==706== Copyright (C) 2002-2017,and GNU GPL'd,by Julian Seward et al.
==706== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==706== Command: ./solution
==706==
==706==
==706== HEAP SUMMARY:
==706==     in use at exit: 122,880 bytes in 6 blocks
==706==   total heap usage: 333,346 allocs,333,340 frees,49,696,751 bytes allocated
==706==
==706== LEAK SUMMARY:
==706==    definitely lost: 0 bytes in 0 blocks
==706==    indirectly lost: 0 bytes in 0 blocks
==706==      possibly lost: 0 bytes in 0 blocks
==706==    still reachable: 122,880 bytes in 6 blocks
==706==         suppressed: 0 bytes in 0 blocks
==706== Rerun with --leak-check=full to see details of leaked memory
==706==
==706== For counts of detected and suppressed errors,rerun with: -v
==706== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Видно, что программа запросила почти 50 миллионов байт из кучи, и при этом не освободила 6 блоков. Это происходит в следующем месте кода:

```
1 //
2 // main.cpp
3 //
4 ...
5     std::ios::sync_with_stdio(false);
6     std::cin.tie(nullptr);
7     std::cout.tie(nullptr);
8 ...
```

На самом деле, согласно [1], это не считается утечкой как таковой. На чекере данная проблема ошибкой выполнения не считается.

3 Gprof

Gprof - средство профилирования в Unix системах. Используется для измерения времени работы отдельных функций программы и общего времени работы программы. Для использования утилиты gprof нужно скомпилировать программу с флагами -pg -g и выполнить её. После выполнения в рабочей директории создастся файл gmon.out. Потом необходимо вызвать утилиту с исполняемым файлом и файлом gmon.out В моем случае профилировка проводится на тесте со стандартными командами, а также с командами загрузки и сохранения дерева. Сначала выводится таблица, в которой показано, сколько времени было затрачено в каждой из вызванных функций.

```
kng@Legion:/mnt/c/vsc/da/lab2/prof$ gprof solution test.out
Flat profile:

Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  us/call  us/call  name
25.02      0.02     0.02     1226    16.32    16.32  NStd::TTree::DeleteTree(NStd::TN
25.02      0.04     0.02     1201    16.66    16.66  NStd::TTree::SaveHelper(NStd::TN
12.51      0.05     0.01    91598     0.11     0.11  NStd::StringCompare(char
const*,char const*)
12.51      0.06     0.01     1225     8.17     8.17  NStd::TTree::LoadHelper(_IO_FILE*
12.51      0.07     0.01                             frame_dummy
12.51      0.08     0.01                             main
0.00       0.08     0.00   724665     0.00     0.00  NStd::TNode::~TNode()
0.00       0.08     0.00   722150     0.00     0.00  NStd::TNode::TNode(char*,unsigned
long long,short)
0.00       0.08     0.00     2554     0.00     0.98  NStd::TTree::Delete(char*)
0.00       0.08     0.00     2554     0.00     0.98  NStd::TTree::Delete(char*,NStd::TN
0.00       0.08     0.00     2515     0.00     0.00  NStd::TNode::TNode(char*,unsigned
long long)
0.00       0.08     0.00     2515     0.00     0.98  NStd::TTree::Insert(char*,unsigned
long long)
0.00       0.08     0.00     2514     0.00     0.98  NStd::TTree::Insert(char*,unsigned
long long,NStd::TNode*)
0.00       0.08     0.00     2503     0.00     0.92  NStd::TTree::Search(char*)
0.00       0.08     0.00     2503     0.00     0.92  NStd::TTree::Search(char*,NStd::TN
0.00       0.08     0.00     1241     0.00     0.00  NStd::TTree::Balance(NStd::TNode*)
0.00       0.08     0.00     1225     0.00    24.49  NStd::TTree::Load(char*)
0.00       0.08     0.00     1201     0.00    16.66  NStd::TTree::Save(char*)
0.00       0.08     0.00      938     0.00     0.00  NStd::TTree::RotateLeft(NStd::TNod
```

```
0.00      0.08     0.00      921     0.00     0.00  NStd::TTree::RotateRight(NStd::TN
0.00      0.08     0.00      323     0.00     0.00  NStd::TTree::BigLeftRotation(NStd
0.00      0.08     0.00      295     0.00     0.00  NStd::TTree::BigRightRotation(NSt
0.00      0.08     0.00        1     0.00     0.00  _GLOBAL__sub_I__ZN4NStd13StringCo
0.00      0.08     0.00        1     0.00     0.00  __static_initialization_and_destr
0.00      0.08     0.00        1     0.00     0.00  NStd::TTree::TTree()
0.00      0.08     0.00        1     0.00    16.32  NStd::TTree::~TTree()
```

Исходя из данных в таблице, можно понять, что больше всего времени ушло на удаление дерева, сохранение, загрузку и сравнение ключей. Удаление дерева вызывается каждый раз, когда необходимо загрузить дерево из файла. Можно увидеть, что трата времени на функции вставки, удаления или поиска очень мала в сравнении с функциями сохранения и загрузки.
Дальше выводится граф вызовов.

```
Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) for 12.49% of 0.08 seconds

index % time    self  children    called     name
<spontaneous>
[1]     87.5    0.01    0.06                 main [1]
0.00    0.03    1225/1225        NStd::TTree::Load(char*) [2]
0.00    0.02    1201/1201        NStd::TTree::Save(char*) [5]
0.00    0.00   24836/91598       NStd::StringCompare(char const*,char const*)
[7]
0.00    0.00    2554/2554        NStd::TTree::Delete(char*) [9]
0.00    0.00    2515/2515        NStd::TTree::Insert(char*,unsigned long long)
[11]
0.00    0.00    2503/2503        NStd::TTree::Search(char*) [13]
0.00    0.00       1/1           NStd::TTree::~TTree() [15]
0.00    0.00       1/1           NStd::TTree::TTree() [31]
-----------------------------------------------
0.00    0.03    1225/1225        main [1]
[2]     37.5    0.00    0.03    1225         NStd::TTree::Load(char*) [2]
0.02    0.00    1225/1226        NStd::TTree::DeleteTree(NStd::TNode*) [3]
0.01    0.00    1225/1225        NStd::TTree::LoadHelper(_IO_FILE*) [8]
-----------------------------------------------
1449330             NStd::TTree::DeleteTree(NStd::TNode*) [3]
0.00    0.00       1/1226        NStd::TTree::~TTree() [15]
0.02    0.00    1225/1226        NStd::TTree::Load(char*) [2]
```

```
[3]     25.0    0.02    0.00    1226+1449330 NStd::TTree::DeleteTree(NStd::TNode*)
[3]
0.00    0.00  724665/724665      NStd::TNode::~TNode() [21]
1449330             NStd::TTree::DeleteTree(NStd::TNode*) [3]
-----------------------------------------------
1345794             NStd::TTree::SaveHelper(NStd::TNode*,_IO_FILE*) [4]
0.02    0.00    1201/1201        NStd::TTree::Save(char*) [5]
[4]     25.0    0.02    0.00    1201+1345794 NStd::TTree::SaveHelper(NStd::TNode*,_IO
[4]
1345794             NStd::TTree::SaveHelper(NStd::TNode*,_IO_FILE*) [4]
-----------------------------------------------
0.00    0.02    1201/1201        main [1]
[5]     25.0    0.00    0.02    1201         NStd::TTree::Save(char*) [5]
0.02    0.00    1201/1201        NStd::TTree::SaveHelper(NStd::TNode*,_IO_FILE*)
[4]
-----------------------------------------------
<spontaneous>
[6]     12.5    0.01    0.00                 frame_dummy [6]
-----------------------------------------------
0.00    0.00   21170/91598       NStd::TTree::Search(char*,NStd::TNode*) [14]
0.00    0.00   22665/91598       NStd::TTree::Insert(char*,unsigned long long,NStd::TN
[12]
0.00    0.00   22927/91598       NStd::TTree::Delete(char*,NStd::TNode*) [10]
0.00    0.00   24836/91598       main [1]
[7]     12.5    0.01    0.00   91598         NStd::StringCompare(char const*,char
const*) [7]
-----------------------------------------------
1444300             NStd::TTree::LoadHelper(_IO_FILE*) [8]
0.01    0.00    1225/1225        NStd::TTree::Load(char*) [2]
[8]     12.5    0.01    0.00    1225+1444300 NStd::TTree::LoadHelper(_IO_FILE*)
[8]
0.00    0.00  722150/722150      NStd::TNode::TNode(char*,unsigned long long,short)
[22]
1444300             NStd::TTree::LoadHelper(_IO_FILE*) [8]
-----------------------------------------------
0.00    0.00    2554/2554        main [1]
[9]      3.1    0.00    0.00    2554         NStd::TTree::Delete(char*) [9]
0.00    0.00    2554/2554        NStd::TTree::Delete(char*,NStd::TNode*) [10]
-----------------------------------------------
22927              NStd::TTree::Delete(char*,NStd::TNode*) [10]
0.00    0.00    2554/2554        NStd::TTree::Delete(char*) [9]
```

```
[10]     3.1    0.00    0.00    2554+22927   NStd::TTree::Delete(char*,NStd::TNode*)
[10]
0.00    0.00   22927/91598       NStd::StringCompare(char const*,char const*)
[7]
22927             NStd::TTree::Delete(char*,NStd::TNode*) [10]
-----------------------------------------------
0.00    0.00    2515/2515        main [1]
[11]     3.1    0.00    0.00    2515         NStd::TTree::Insert(char*,unsigned
long long) [11]
0.00    0.00    2514/2514        NStd::TTree::Insert(char*,unsigned long long,NStd::TN
[12]
0.00    0.00       1/2515        NStd::TNode::TNode(char*,unsigned long long)
[23]
-----------------------------------------------
22665             NStd::TTree::Insert(char*,unsigned long long,NStd::TNode*)
[12]
0.00    0.00    2514/2514        NStd::TTree::Insert(char*,unsigned long long)
[11]
[12]     3.1    0.00    0.00    2514+22665   NStd::TTree::Insert(char*,unsigned
long long,NStd::TNode*) [12]
0.00    0.00   22665/91598       NStd::StringCompare(char const*,char const*)
[7]
0.00    0.00    2514/2515        NStd::TNode::TNode(char*,unsigned long long)
[23]
0.00    0.00    1241/1241        NStd::TTree::Balance(NStd::TNode*) [24]
22665             NStd::TTree::Insert(char*,unsigned long long,NStd::TNode*)
[12]
-----------------------------------------------
0.00    0.00    2503/2503        main [1]
[13]     2.9    0.00    0.00    2503         NStd::TTree::Search(char*) [13]
0.00    0.00    2503/2503        NStd::TTree::Search(char*,NStd::TNode*) [14]
-----------------------------------------------
19938             NStd::TTree::Search(char*,NStd::TNode*) [14]
0.00    0.00    2503/2503        NStd::TTree::Search(char*) [13]
[14]     2.9    0.00    0.00    2503+19938   NStd::TTree::Search(char*,NStd::TNode*)
[14]
0.00    0.00   21170/91598       NStd::StringCompare(char const*,char const*)
[7]
19938             NStd::TTree::Search(char*,NStd::TNode*) [14]
-----------------------------------------------
0.00    0.00       1/1           main [1]
```

```
[15]     0.0    0.00    0.00       1         NStd::TTree::~TTree() [15]
0.00    0.00       1/1226        NStd::TTree::DeleteTree(NStd::TNode*) [3]
-----------------------------------------------
0.00    0.00  724665/724665      NStd::TTree::DeleteTree(NStd::TNode*) [3]
[21]     0.0    0.00    0.00  724665         NStd::TNode::~TNode() [21]
-----------------------------------------------
0.00    0.00  722150/722150      NStd::TTree::LoadHelper(_IO_FILE*) [8]
[22]     0.0    0.00    0.00  722150         NStd::TNode::TNode(char*,unsigned
long long,short) [22]
-----------------------------------------------
0.00    0.00       1/2515        NStd::TTree::Insert(char*,unsigned long long)
[11]
0.00    0.00    2514/2515        NStd::TTree::Insert(char*,unsigned long long,NStd::TN
[12]
[23]     0.0    0.00    0.00    2515         NStd::TNode::TNode(char*,unsigned
long long) [23]
-----------------------------------------------
0.00    0.00    1241/1241        NStd::TTree::Insert(char*,unsigned long long,NStd::TN
[12]
[24]     0.0    0.00    0.00    1241         NStd::TTree::Balance(NStd::TNode*)
[24]
0.00    0.00     323/323         NStd::TTree::BigLeftRotation(NStd::TNode*)
[27]
0.00    0.00     320/938         NStd::TTree::RotateLeft(NStd::TNode*) [25]
0.00    0.00     303/921         NStd::TTree::RotateRight(NStd::TNode*) [26]
0.00    0.00     295/295         NStd::TTree::BigRightRotation(NStd::TNode*)
[28]
-----------------------------------------------
0.00    0.00     295/938         NStd::TTree::BigRightRotation(NStd::TNode*)
[28]
0.00    0.00     320/938         NStd::TTree::Balance(NStd::TNode*) [24]
0.00    0.00     323/938         NStd::TTree::BigLeftRotation(NStd::TNode*)
[27]
[25]     0.0    0.00    0.00     938         NStd::TTree::RotateLeft(NStd::TNode*)
[25]
-----------------------------------------------
0.00    0.00     295/921         NStd::TTree::BigRightRotation(NStd::TNode*)
[28]
0.00    0.00     303/921         NStd::TTree::Balance(NStd::TNode*) [24]
0.00    0.00     323/921         NStd::TTree::BigLeftRotation(NStd::TNode*)
[27]
```

```
[26]     0.0    0.00    0.00     921         NStd::TTree::RotateRight(NStd::TNode*)
[26]
-----------------------------------------------
0.00    0.00     323/323         NStd::TTree::Balance(NStd::TNode*) [24]
[27]     0.0    0.00    0.00     323         NStd::TTree::BigLeftRotation(NStd::TNode*
[27]
0.00    0.00     323/921         NStd::TTree::RotateRight(NStd::TNode*) [26]
0.00    0.00     323/938         NStd::TTree::RotateLeft(NStd::TNode*) [25]
-----------------------------------------------
0.00    0.00     295/295         NStd::TTree::Balance(NStd::TNode*) [24]
[28]     0.0    0.00    0.00     295         NStd::TTree::BigRightRotation(NStd::TNode
[28]
0.00    0.00     295/938         NStd::TTree::RotateLeft(NStd::TNode*) [25]
0.00    0.00     295/921         NStd::TTree::RotateRight(NStd::TNode*) [26]
-----------------------------------------------
0.00    0.00       1/1           __libc_csu_init [38]
[29]     0.0    0.00    0.00       1         _GLOBAL__sub_I__ZN4NStd13StringCompareEPK
[29]
0.00    0.00       1/1           __static_initialization_and_destruction_0(int,int)
[30]
-----------------------------------------------
0.00    0.00       1/1           _GLOBAL__sub_I__ZN4NStd13StringCompareEPKcS1_
[29]
[30]     0.0    0.00    0.00       1         __static_initialization_and_destruction_0
[30]
-----------------------------------------------
0.00    0.00       1/1           main [1]
[31]     0.0    0.00    0.00       1         NStd::TTree::TTree() [31]
-----------------------------------------------
```

Граф вызовов позволяет понять откуда и сколько раз были вызваны функции.

4 Выводы

Подводя итоги, хочется сказать про найденные мной недочёты. После профилировки я удивился, насколько долго работают функции сохранения и загрузки, и попытался ускорить их. Мне удалось добиться ускорения примерно в 5 раз, заменив работу с потоками на работу с файлами из C. Также, я заметил, что в функциях вставки, удаления и поиска неоптимально используется функция сравнения ключей. Она вызывалась по 3 раза последовательно, что на самом деле можно было заменить одним вызовом в начале функции и последующим сравнением с результатом. Также, в функции main находился страшный цикл:

```
for(int i = 0; i < strlen(key); ++i) {
    key[i] = (char)tolower(key[i]);
}
```

Который был заменен на:

```
keyLen = strlen(key);
for(int i = 0; i < keyLen; ++i) {
    key[i] = (char)tolower(key[i]);
}
```

Дело в том, что функция strlen вызывалась много раз, что приводило к значительному замедлению работы программы. Хочется сказать, что получить вердикт "OK"без использования valgrind'a и gprof'a сильно сложнее, чем с ними.

Список литературы

[1] *Stackoverflow*
URL: `https://stackoverflow.com/questions/59104808/getting-a-still-reachable-in-leak`
(дата обращения: 27.11.2020).