Lab 5 report

Ethan Coe-Renner

October 1, 2021

Contents

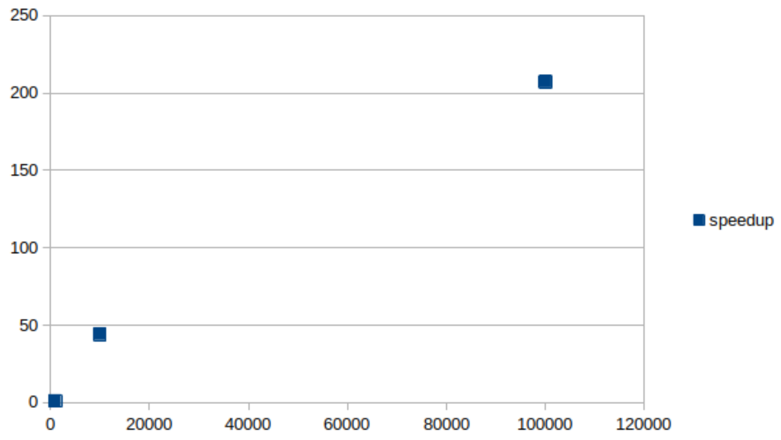
1.

	Bubble	Tree
Real	29	0.155
User	29	0.15
System	0.00	0.00

2. swapping/assignment takes user time, as well as comparison

3. memory allocation takes kernal time

4. The tree sort is the fastest.



5. using array size of 25000

	IR
Tree	21677878
Bubble	9081667095

6.

	1st most active	2nd most active	3rd most active
Bubble	`bubbleSort`	`random_r`	`random`
Tree	`insert_element'2`	`_int_malloc`	`_int_free`

7.

	most cpu intensive line
Bubble	`if (array_start[j-1] > array_start[j])`
Tree	`struct BTreeNode *newNode = malloc(sizeof(struct BTreeNode))`

8. Valgrind output for mergesort with intentional memory leak

```
==26336== Memcheck, a memory error detector
==26336== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==26336== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==26336== Command: ./sorting_program merge
==26336== Parent PID: 15670
==26336==
==26336==
==26336== HEAP SUMMARY:
==26336==     in use at exit: 100,000 bytes in 1 blocks
==26336==   total heap usage: 2 allocs, 1 frees, 101,024 bytes allocated
==26336==
==26336== 100,000 bytes in 1 blocks are definitely lost in loss record 1 of 1
==26336==    at 0x483A971: calloc (in /nix/store/hn8gpd3jfg5dm6hk9xaqhshxc5nzvvj9-
==26336==    by 0x401283: main (sorting.c:42) // This line indicates the original
==26336==
==26336== LEAK SUMMARY:
==26336==    definitely lost: 100,000 bytes in 1 blocks
==26336==    indirectly lost: 0 bytes in 0 blocks
==26336==      possibly lost: 0 bytes in 0 blocks
==26336==    still reachable: 0 bytes in 0 blocks
==26336==         suppressed: 0 bytes in 0 blocks
==26336==
==26336== For lists of detected and suppressed errors, rerun with: -s
==26336== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

9. 100000 bytes were leaked in the buggy program

10. Valgrind output for mergesort with intentional memory leak fixed

```
==26538== Memcheck, a memory error detector
```

```
==26538== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==26538== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==26538== Command: ./sorting_program merge
==26538== Parent PID: 15670
==26538==
==26538==
==26538== HEAP SUMMARY:
==26538==     in use at exit: 0 bytes in 0 blocks
==26538==   total heap usage: 2 allocs, 2 frees, 101,024 bytes allocated
==26538==
==26538== All heap blocks were freed -- no leaks are possible
==26538==
==26538== For lists of detected and suppressed errors, rerun with: -s
==26538== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

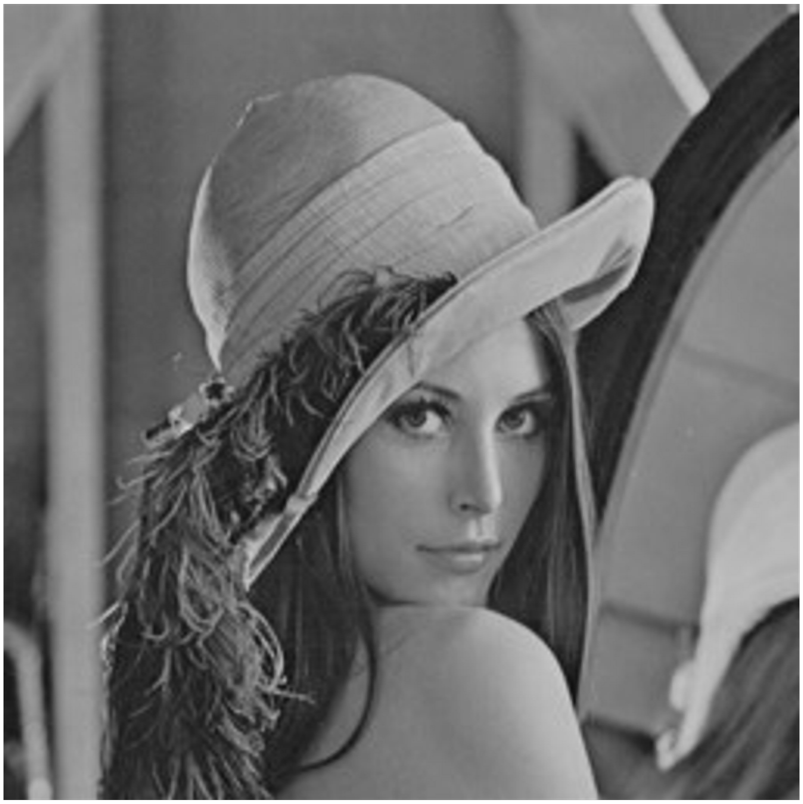
11. Valgrind output for tree sort

    ```
    ==26594== Memcheck, a memory error detector
    ==26594== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
    ==26594== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
    ==26594== Command: ./sorting_program tree
    ==26594== Parent PID: 15670
    ==26594==
    ==26594==
    ==26594== HEAP SUMMARY:
    ==26594==     in use at exit: 0 bytes in 0 blocks
    ==26594==   total heap usage: 25,001 allocs, 25,001 frees, 601,024 bytes allocated
    ==26594==
    ==26594== All heap blocks were freed -- no leaks are possible
    ==26594==
    ==26594== For lists of detected and suppressed errors, rerun with: -s
    ==26594== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
    ```

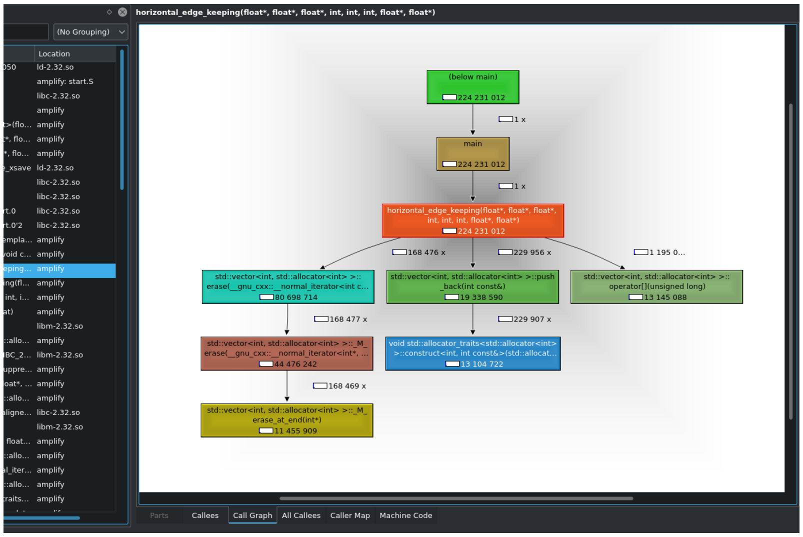
12. `time ./amplify IMAGES/Lenna_org_1024.pgm 11 1.1 2`

    2.31 user 0.06 system 2.438 total

13. When the computer has multiple threads, system time will not be an actual time, but the sum of the time taken by each thread.

14. 

15. top 3 functions: `convolve`, `mean_keeping`, `double_threshold`

16. 

horizontal_edge_keeping(float*, float*, float*, int, int, int, float*, float*)

	Location
050	ld-2.32.so
	amplify: start.S
	libc-2.32.so
	amplify
t>(flo…	amplify
*, flo…	amplify
*, flo…	amplify
e_xsave	ld-2.32.so
	libc-2.32.so
	libc-2.32.so
rt.0	libc-2.32.so
rt.0'2	libc-2.32.so
empla…	amplify
void c…	amplify
eping…	amplify
ing(fl…	amplify
int, i…	amplify
at)	amplify
	libm-2.32.so
::allo…	amplify
IBC_2…	libm-2.32.so
uppre…	amplify
oat*, …	amplify
::allo…	amplify
aligne…	libc-2.32.so
	libm-2.32.so
float…	amplify
::allo…	amplify
al_iter…	amplify
::allo…	amplify
raits…	amplify

(No Grouping)
(below main)
224 231 012
1 x
main
224 231 012
1 x
horizontal_edge_keeping(float*, float*, float*, int, int, int, float*, float*)
224 231 012
168 476 x
229 956 x
1 195 0…
std::vector<int, std::allocator<int> >::erase(__gnu_cxx::__normal_iterator<int c…
80 698 714
std::vector<int, std::allocator<int> >::push_back(int const&)
19 338 590
std::vector<int, std::allocator<int> >::operator[](unsigned long)
13 145 088
168 477 x
229 907 x
std::vector<int, std::allocator<int> >::_M_erase(__gnu_cxx::__normal_iterator<int*, …
44 476 242
void std::allocator_traits<std::allocator<int> >::construct<int, int const&>(std::allocat…
13 104 722
168 469 x
std::vector<int, std::allocator<int> >::_M_erase_at_end(int*)
11 455 909
Parts
Callees
Call Graph
All Callees
Caller Map
Machine Code

17. convolve occupies 76% of the execution time

18. yes, valgrind shows that it lost 16,778,236 bytes in 8 blocks