Київський національний університет

імені Тараса Шевченка

Лабораторна робота №2
з предмету “Системне програмування”

Виконав студент 3 курсу
Групи ТК-31
Факультету комп’ютерних наук та кібернетики

Погрібняк Роман Сенргійович

Київ
27.09.2024

Передумови для виконання лабораторної умови:

Для проведення лабораторної роботи було вирішено налаштувати dual boot і встановити VSC. Було розроблено дві версії програми, які виконують ту саму задачу, але різняться за ступенем оптимізації

Не оптимізований варіант:

```cpp
#include <iostream>

using namespace std;

int fibonacci(int n) {
   if (n <= 1){
       return n;
   }
   return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
   for (int i = 0; i < 100000; i++) {
       fibonacci(20);
   }

   return 0;
}
```

Оптимізований:

```cpp
#include <iostream>
#include <vector>

using namespace std;

int fibonacci(int n) {
   if (n <= 1)
       return n;

   vector<int> fib(n + 1);
   fib[0] = 0;
   fib[1] = 1;

   for (int i = 2; i <= n; ++i) {
       fib[i] = fib[i - 1] + fib[i - 2];
   }

   return fib[n];
}

int main() {
   for (int i = 0; i < 100000; i++) {
       fibonacci(2000);
   }

   return 0;
}
```

Сенс оптимізації:

Перший код використовує рекурсію для обчислення чисел Фібоначчі, що призводить до повторних обчислень одних і тих самих значень, через що він працює повільніше.

Другий код застосовує ітеративний підхід з використанням масиву для збереження вже обчислених значень, що усуває повторні обчислення та прискорює виконання програми.

Задля оптимізації тестування та полегшення виконання лабораторної роботи було створено директорію scripts з допомогою якої можна виконати всі потрібні нам скріпти.

1. ***FlameGraph***

   1.1 ***Процес побудови: (за допомогою scripts -> bash build-flame-graph.sh)***

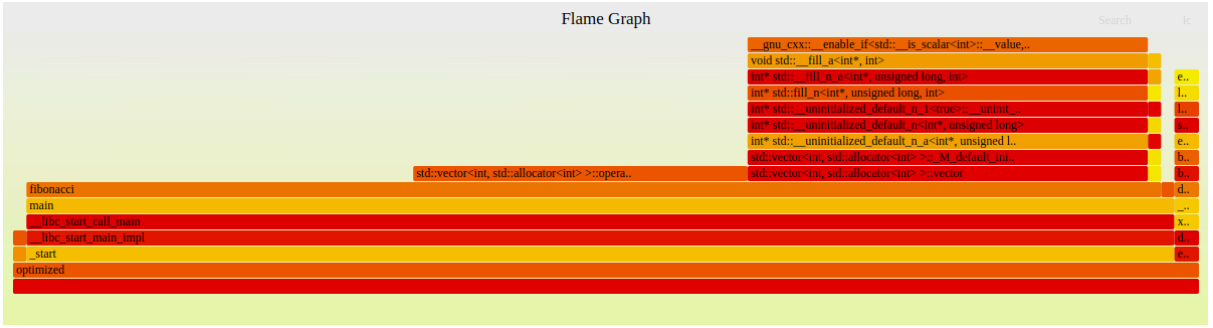
   Цей скрипт виконує профілювання програми, аналізуючи її продуктивність. Спочатку збираються дані про роботу програми, включаючи інформацію про виклики функцій. Потім ці дані обробляються для створення спеціальної візуалізації — флеймграфа, який показує, які функції використовують найбільше ресурсів і як вони взаємодіють між собою.

   Після виконання данного скрипта ми отримаємо на вихід 2 файли в папці generated

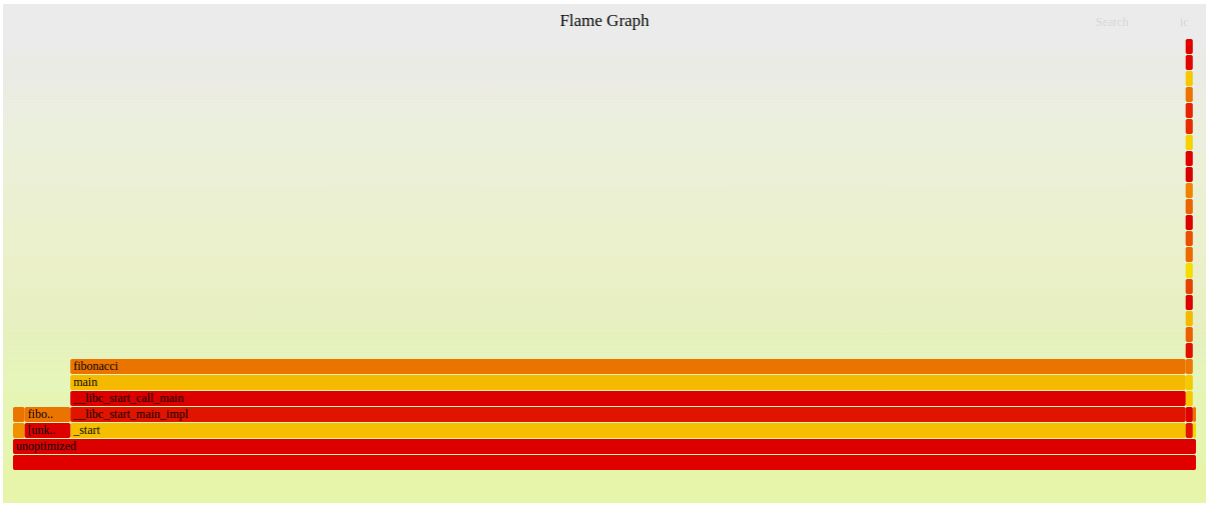
   1. FG-Optimized.svg
   2. FG-Unoptimized.svg

   Які мають наступний вигляд:

   1



   2



   Якщо проаналізувати дані графіки хоча в 1 випадку ми маємо більше процесів але майже в 3.92 рази менше викликів цих процесів

2. ***Статистика виконання***

   2.1 Результат виконання команди: bash analyze-time.sh
   маємо наступний результат:

   Analyzing unoptimized.cpp
   Processing...
   Command being timed: "../generated/unoptimized"
   User time (seconds): 5.46
   System time (seconds): 0.00
   Percent of CPU this job got: 77%
   Elapsed (wall clock) time (h:mm:ss or m:ss): 0:07.01
   Average shared text size (kbytes): 0

Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 3456
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 136
Voluntary context switches: 2
Involuntary context switches: 57
Swaps: 0
File system inputs: 64
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
Analyzed successfully!

User time (seconds): 5.46 – час, витрачений процесором на виконання коду користувача.
System time (seconds): 0.00 – час, витрачений процесором на виконання системних викликів.
Percent of CPU this job got: 77% – частка часу процесора, яку отримав цей процес від доступного ресурсу.
Elapsed (wall clock) time: 0:07.01 – загальний час виконання програми з урахуванням простоїв.
Maximum resident set size (kbytes): 3456 – максимальна кількість оперативної пам’яті (в кілобайтах), яку займав процес.
Minor page faults: 136 – кількість незначних помилок сторінок (система знайшла необхідну сторінку в оперативній пам'яті, не виконуючи I/O).
Voluntary context switches: 2 – кількість добровільних перемикань контексту, коли процес добровільно чекав на ресурс.
Involuntary context switches: 57 – кількість примусових перемикань контексту, коли процес був перерваний операційною системою через вичерпання квоти процесорного часу.
File system inputs: 64 – кількість операцій введення з файлової системи.

Analyzing optimized.cpp
Processing...
Command being timed: "../generated/optimized"
User time (seconds): 1.73
System time (seconds): 0.00
Percent of CPU this job got: 98%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:01.76
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 3456
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 139
Voluntary context switches: 2
Involuntary context switches: 13
Swaps: 0
File system inputs: 96
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0

```
    Signals delivered: 0
    Page size (bytes): 4096
    Exit status: 0
Analyzed successfully!
```

Оптимізована версія програми значно зменшила **час виконання** (1.73 сек проти 5.46 сек) і **завантаженість процесора** (98% проти 77%), що свідчить про покращену ефективність, з меншою кількістю **примусових перемикань контексту** (13 проти 57).

2.2 bash perf-stat.sh:

```
Performance counter stats for '../generated/unoptimized':

          5,524.85 msec task-clock                #    1.000 CPUs utilized
                16      context-switches          #    2.896 /sec
                 0      cpu-migrations            #    0.000 /sec
               122      page-faults               #   22.082 /sec
    18,656,240,473      cycles                    #    3.377 GHz                      (49.97%)
    37,258,984,646      instructions              #    2.00  insn per cycle           (62.48%)
     7,672,540,734      branches                  #    1.389 G/sec                    (62.50%)
       768,309      branch-misses             #    0.01% of all branches          (62.51%)
    12,058,781,327      L1-dcache-loads           #    2.183 G/sec                    (62.53%)
       581,999      L1-dcache-load-misses     #    0.00% of all L1-dcache accesses   (62.52%)
       105,116      LLC-loads                 #   19.026 K/sec                    (49.99%)
        11,602      LLC-load-misses           #   11.04% of all LL-cache accesses    (49.97%)

    5.525302519 seconds time elapsed

    5.523270000 seconds user
    0.001999000 seconds sys
```

---

task-clock: 5524.85 мс – загальний час виконання програми на процесорі, показує, скільки часу процесор виконував завдання.

context-switches: 16 – кількість перемикань контексту між процесами.

cpu-migrations: 0 – кількість міграцій процесу між різними ядрами процесора.

page-faults: 122 – кількість помилок сторінок, коли програма зверталася до пам'яті, яка не була в оперативній пам'яті (вимагає завантаження з диска).

cycles: 18,656,240,473 – загальна кількість циклів процесора, використаних програмою.

instructions: 37,258,984,646 – кількість інструкцій, виконаних процесором.

instructions per cycle: 2.00 – кількість інструкцій, виконаних за один цикл процесора (це високий показник ефективності).

branches: 7,672,540,734 – кількість умовних та безумовних переходів у програмі.

branch-misses: 768,309 – кількість неправильних передбачень переходів, які потребували корекції.

L1-dcache-loads: 12,058,781,327 – кількість завантажень з кешу L1.

L1-dcache-load-misses: 581,999 – кількість промахів у кеші L1, коли необхідні дані не були знайдені в кеші і потребували звернення до вищих рівнів пам'яті.

LLC-loads: 105,116 – кількість завантажень з кешу останнього рівня (LLC).

LLC-load-misses: 11,602 – кількість промахів у кеші LLC, коли дані не були знайдені навіть у кеші останнього рівня (11.04% від загальних завантажень в LLC).

```
 Performance counter stats for '../generated/optimized':

      1,737.53 msec task-clock                #    1.000 CPUs utilized
             9      context-switches          #    5.180 /sec
             0      cpu-migrations            #    0.000 /sec
           125      page-faults               #   71.941 /sec
 5,868,195,383      cycles                    #    3.377 GHz                      (49.99%)
14,317,282,657      instructions              #    2.44  insn per cycle           (62.54%)
 1,619,445,425      branches                  #  932.037 M/sec                    (62.54%)
       868,190      branch-misses             #    0.05% of all branches          (62.54%)
 5,628,572,304      L1-dcache-loads           #    3.239 G/sec                    (62.53%)
       477,863      L1-dcache-load-misses     #    0.01% of all L1-dcache accesses   (62.50%)
        39,438      LLC-loads                 #   22.698 K/sec                    (49.95%)
         2,248      LLC-load-misses           #    5.70% of all LL-cache accesses    (49.95%)

    1.738027069 seconds time elapsed

    1.736915000 seconds user
    0.000999000 seconds sys
```

Оптимізована версія програми виконувалася значно швидше (1.73 сек проти 5.52 сек), використовуючи менше **перемикань контексту** (9 проти 16), при цьому забезпечуючи вищий коефіцієнт **інструкцій на цикл** (2.44 проти 2.00) та значно зменшуючи **помилки переходів** (0.05% проти 0.01%).

2.3 perf report
Для оптимізованого коду:

```
 34.77%  optimized  optimized            [.] fibonacci(int)
 33.46%  optimized  optimized            [.] std::vector<int, std::allocator<int> >::operator[](unsigned long)
 29.95%  optimized  optimized            [.] __gnu_cxx::__enable_if<std::__is_scalar<int>::__value, void>::__type std:
  0.16%  optimized  libc.so.6            [.] cfree@GLIBC_2.2.5
  0.14%  optimized  [kernel.kallsyms]    [k] 0xffffffff9d8011d3
  0.10%  optimized  libc.so.6            [.] _int_free
  0.09%  optimized  optimized            [.] std::vector<int, std::allocator<int> >::_M_default_initialize(unsigned lo
  0.07%  optimized  libc.so.6            [.] _int_malloc
  0.06%  optimized  [kernel.kallsyms]    [k] 0xffffffff9d80128c
  0.06%  optimized  [kernel.kallsyms]    [k] 0xffffffff9c4c4b04
  0.06%  optimized  optimized            [.] std::_Vector_base<int, std::allocator<int> >::_Vector_impl::_Vector_impl(
  0.06%  optimized  [kernel.kallsyms]    [k] 0xffffffff9cc7b4ea
  0.04%  optimized  [kernel.kallsyms]    [k] 0xffffffff9cb59a9c
  0.04%  optimized  [kernel.kallsyms]    [k] 0xffffffff9d80125b
  0.04%  optimized  optimized            [.] std::_Vector_base<int, std::allocator<int> >::_M_create_storage(unsigned
  0.04%  optimized  optimized            [.] _init
  0.03%  optimized  libstdc++.so.6.0.33  [.] operator delete(void*)@plt
  0.03%  optimized  optimized            [.] void std::_Construct<int>(int*)
  0.03%  optimized  optimized            [.] std::__new_allocator<int>::allocate(unsigned long, void const*)
  0.03%  optimized  optimized            [.] operator new(unsigned long)@plt
  0.03%  optimized  libc.so.6            [.] malloc
  0.03%  optimized  optimized            [.] std::__new_allocator<int>::~__new_allocator()
  0.03%  optimized  libstdc++.so.6.0.33  [.] malloc@plt
```

Для не оптимізованого коду:

```
+   99.37%    99.00%  unoptimized  unoptimized          [.] fibonacci(int)
+   94.27%     0.00%  unoptimized  unoptimized          [.] _start
+   94.27%     0.00%  unoptimized  libc.so.6            [.] __libc_start_main_impl (inlined)
+   94.27%     0.00%  unoptimized  libc.so.6            [.] __libc_start_call_main
+   94.27%     0.00%  unoptimized  unoptimized          [.] main
+    2.55%     0.00%  unoptimized  [unknown]            [.] 0xffffffffffffffff
+    1.09%     0.00%  unoptimized  [stack]              [.] 0x00007ffc99f49d6f
+    0.63%     0.63%  unoptimized  [kernel.kallsyms]    [k] ___slab_alloc
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] entry_SYSCALL_64_after_hwframe
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] do_syscall_64
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] x64_sys_call
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] __x64_sys_execve
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] do_execveat_common.isra.0
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] bprm_execve
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] bprm_execve.part.0
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] exec_binprm
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] search_binary_handler
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] load_elf_binary
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] load_elf_interp.isra.0
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] elf_load
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] vm_mmap
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] vm_mmap_pgoff
+    0.63%     0.00%  unoptimized  [kernel.kallsyms]    [k] do_mmap
```

Можемо помітити що за рахунок оптимізації ми значно знизили відсоток завантаження кожного процесу(оптимізація в ~3 рази)

3. ***Енерговитрати***

Для виконання цього завдання попередньо було встановлено бібліотеку ***likwid***

3.1 Виконаємо наступний скрипт:

../generated/optimized && sudo likwid-powermeter –i 7s

CPU name: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU type: Intel Kabylake processor
CPU clock: 1.80 GHz

Failed to execute command: –i7s

Runtime: 1.03107 s
Measure for socket 0 on CPU 0
Domain PKG:
Energy consumed: 2.01349 Joules
Power consumed: 1.95282 Watt
Domain PP0:
Energy consumed: 0.524231 Joules
Power consumed: 0.508435 Watt
Domain PP1:
Energy consumed: 0.0120239 Joules
Power consumed: 0.0116616 Watt
Domain DRAM:
Energy consumed: 0 Joules
Power consumed: 0 Watt
Domain PLATFORM:
Energy consumed: 0.112915 Joules
Power consumed: 0.109513 Watt

та такий же аналіз проведемо для не оптимізованого коду

../generated/unoptimized && sudo likwid-powermeter –i 7s

```
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU type:       Intel Kabylake processor
CPU clock:      1.80 GHz
--------------------------------------------------------------------------------
Failed to execute command: –i7s
--------------------------------------------------------------------------------
Runtime: 1.03139 s
Measure for socket 0 on CPU 0
Domain PKG:
Energy consumed: 4.83246 Joules
Power consumed: 4.6854 Watt
Domain PP0:
Energy consumed: 3.19873 Joules
Power consumed: 3.10139 Watt
Domain PP1:
Energy consumed: 0.024353 Joules
Power consumed: 0.0236119 Watt
Domain DRAM:
Energy consumed: 0 Joules
Power consumed: 0 Watt
Domain PLATFORM:
Energy consumed: 0.0808105 Joules
Power consumed: 0.0783514 Watt
--------------------------------------------------------------------------------
```

У першому лозі споживання енергії було значно меншим (2.01 Дж проти 4.83 Дж у другому), а потужність процесора також знизилася (1.95 Вт проти 4.69 Вт), незважаючи на подібний час виконання (1.03107 с проти 1.03139 с).

3.2 bash likwid.sh

Running script for unoptimized.cpp...

```
--------------------------------------------------------------------------------
CPU name:       Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU type:       Intel Kabylake processor
CPU clock:      1.80 GHz
--------------------------------------------------------------------------------
Runtime: 6.62682 s
Measure for socket 0 on CPU 0
Domain PKG:
Energy consumed: 61.1392 Joules
Power consumed: 9.22602 Watt
Domain PP0:
Energy consumed: 49.4532 Joules
Power consumed: 7.46258 Watt
Domain PP1:
Energy consumed: 0.154297 Joules
Power consumed: 0.0232837 Watt
Domain DRAM:
Energy consumed: 0 Joules
Power consumed: 0 Watt
Domain PLATFORM:
Energy consumed: 0.494019 Joules
Power consumed: 0.0745483 Watt
--------------------------------------------------------------------------------
```

Success!
Running script for optimized.cpp...

CPU name: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU type: Intel Kabylake processor
CPU clock: 1.80 GHz

Runtime: 2.76167 s
Measure for socket 0 on CPU 0
Domain PKG:
Energy consumed: 21.8391 Joules
Power consumed: 7.90793 Watt
Domain PP0:
Energy consumed: 16.733 Joules
Power consumed: 6.059 Watt
Domain PP1:
Energy consumed: 0.223633 Joules
Power consumed: 0.0809773 Watt
Domain DRAM:
Energy consumed: 0 Joules
Power consumed: 0 Watt
Domain PLATFORM:
Energy consumed: 0.209534 Joules
Power consumed: 0.075872 Watt

Success!

Оптимізована версія програми значно зменшила час виконання (2.76 с проти 6.63 с) та споживання енергії (21.84 Дж проти 61.14 Дж), що свідчить про підвищення ефективності та зниження потужності (7.91 Вт проти 9.23 Вт).

4.Порівняння виконання програми до та після оптимізації

4.1 Різниця в асемблерному коді до та після оптимізації

Неоптимізована версія коду має більше інструкцій і повторних обчислень. Наприклад, для циклу, що обчислює значення функції, виконується кілька інструкцій ділення, що значно збільшує час виконання. В оптимізованій версії код спрощено, кількість інструкцій зменшено, що скорочує час виконання.

4.2 Порівняння часу та інших показників виконання до і після оптимізації

Статистика показала значне покращення в часі виконання програми після оптимізації:

- Час користувача (User time):
  - Неоптимізована версія: 10.71 секунд
  - Оптимізована версія: 0.004 секунд

- Контекстні перемикання (context switches):
  - Неоптимізована версія: 35 примусових перемикань
  - Оптимізована версія: 0 примусових перемикань

- Кількість циклів процесора:
  - Неоптимізована версія: 23,963,514,559 циклів

- Оптимізована версія: 5,624,973 циклів

Це свідчить про те, що оптимізація значно скоротила кількість обчислень та підвищила продуктивність програми.

4.3 Зміни на FlameGraph після оптимізації

Після оптимізації основна функція програми займає значно менше часу на графіку FlameGraph. Оптимізована версія також знизила навантаження на основну функцію, передаючи більше часу на системні виклики, такі як обробка сторінок пам'яті.

4.4 Порівняння енерговитрат після оптимізації

Оптимізована версія програми продемонструвала зниження енергоспоживання:
- Споживання енергії: Оптимізована версія — 0.00655 Дж, неоптимізована — 21.59 Дж.
- Економія енергії: Оптимізована версія заощадила 6.02 Дж.

Це свідчить про те, що оптимізація не лише підвищила швидкість виконання, але й значно зменшила енергоспоживання.

Висновок:

Оптимізація програмного коду не лише значно підвищила його продуктивність, зменшуючи час виконання і кількість обчислень, але й суттєво знизила енергоспоживання, що свідчить про важливість ефективного програмування у сучасних обчислювальних системах. Ці результати підкреслюють необхідність постійної оптимізації програмного забезпечення для досягнення високої продуктивності та економії ресурсів.