Київський національний університет

імені Тараса Шевченка

Лабораторна робота №6

з предмету “Системне програмування”

Виконав

Студент 3 курсу

Групи ТТП-32

Факультету комп’ютерних наук

та кібернетики

Ходаков Максим

Київ

14.11.2023

# Хід роботи

# Генерація FlameGraph:

1. Commands to get a flame graph
2. git clone https://github.com/brendangregg/FlameGraph
3. perf record -F 50 --call-graph dwarf ./main
4. perf script | /home/jovakinn/CLionProjects/flame/FlameGraph/stackcollapse-perf.pl |

/home/jovakinn/CLionProjects/flame/FlameGraph/stackcollapse-recursive.pl |

/home/jovakinn/CLionProjects/flame/FlameGraph/flamegraph.pl > outMain.svg

1. perf record -F 50 --call-graph dwarf ./optimized
2. perf script | /home/jovakinn/CLionProjects/flame/FlameGraph/stackcollapse-perf.pl |

/home/jovakinn/CLionProjects/flame/FlameGraph/stackcollapse-recursive.pl | /home/jovakinn/CLionProjects/flame/FlameGraph/flamegraph.pl > outOptimized.svg

# Команди для статистики /usr/bin/time --verbose ./main

Command being timed: "./main"

User time (seconds): 2.20

System time (seconds): 0.54

Percent of CPU this job got: 6%

Elapsed (wall clock) time (h:mm:ss or m:ss): 0:42.32

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: 1

Minor (reclaiming a frame) page faults: 136

Voluntary context switches: 4523

Involuntary context switches: 1791

Swaps: 0

File system inputs: 32

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0

/usr/bin/time --verbose ./optimized

Command being timed: "./optimized"

User time (seconds): 0.58

System time (seconds): 0.58

Percent of CPU this job got: 2%

Elapsed (wall clock) time (h:mm:ss or m:ss): 0:47.55

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: 1

Minor (reclaiming a frame) page faults: 135

Voluntary context switches: 4523

Involuntary context switches: 1770

Swaps: 0

File system inputs: 32

File system outputs: 0

Socket messages sent: 0

Socket messages received: 0

Signals delivered: 0

Page size (bytes): 4096

Exit status: 0

perf stat -d ./main

Performance counter stats for './main':

2 759,46 msec task-clock # 0,065 CPUs utilized

6 770 context-switches # 2,453 K/sec

0 cpu-migrations # 0,000 /sec

125 page-faults # 45,299 /sec

<not supported> cycles

<not supported> instructions

<not supported> branches

<not supported> branch-misses

<not supported> L1-dcache-loads

<not supported> L1-dcache-load-misses

<not supported> LLC-loads

<not supported> LLC-load-misses

42,451228486 seconds time elapsed

2,157220000 seconds user

0,604571000 seconds sys

perf stat -d ./main

Performance counter stats for './optimized':

 169,81 msec task-clock # 0,032 CPUs utilized

6 575 context-switches # 5,621 K/sec

0 cpu-migrations # 0,000 /sec

125 page-faults # 106,855 /sec

<not supported> cycles

<not supported> instructions

<not supported> branches

<not supported> branch-misses

<not supported> L1-dcache-loads

<not supported> L1-dcache-load-misses

<not supported> LLC-loads

<not supported> LLC-load-misses

36,861763906 seconds time elapsed

0,592626000 seconds user

0,581785000 seconds sys

perf record ./main perf report

Overhead Command Shared Object Symbol

40,57% main main [.] gcd

29,01% main [kernel.kallsyms] [k] finish\_task\_switch.isra.0

7,11% main libc.so.6 [.] \_\_GI\_\_\_libc\_write

2,55% main [kernel.kallsyms] [k] \_raw\_spin\_unlock\_irqrestore

2,25% main [kernel.kallsyms] [k] syscall\_enter\_from\_user\_mode

2,18% main libc.so.6 [.] \_IO\_fwrite

1,50% main [kernel.kallsyms] [k] process\_output\_block

0,96% main [vboxguest] [k] vbg\_req\_perform

0,60% main [kernel.kallsyms] [k] n\_tty\_write

0,58% main libc.so.6 [.] \_IO\_file\_xsputn@@GLIBC\_2.2.5

0,56% main [kernel.kallsyms] [k] file\_tty\_write.constprop.0

0,56% main libstdc++.so.6.0.30 [.] std::num\_put<char, std::ostreambuf\_iterator<char, std::char\_traits<char> > >::\_M\_insert\_int<long>

0,51% main [kernel.kallsyms] [k] pty\_write

0,51% main libc.so.6 [.] \_IO\_fflush

0,49% main [kernel.kallsyms] [k] \_\_check\_heap\_object

0,49% main [kernel.kallsyms] [k] pty\_write\_room

0,45% main [kernel.kallsyms] [k] queue\_work\_on

0,45% main libstdc++.so.6.0.30 [.] std::ostream::put

0,43% main [kernel.kallsyms] [k] vfs\_write

0,39% main [kernel.kallsyms] [k] apparmor\_file\_permission

0,36% main libstdc++.so.6.0.30 [.] std::ostream::sentry::sentry

0,34% main [kernel.kallsyms] [k] \_\_fget\_light

0,32% main libstdc++.so.6.0.30 [.] std::ostream::\_M\_insert<long>

0,30% main [kernel.kallsyms] [k] do\_tty\_write

0,28% main libstdc++.so.6.0.30 [.] std::\_\_ostream\_insert<char, std::char\_traits<char> >

perf record ./optimized perf report

Samples: 2K of event 'cpu-clock:pppH', Event count (approx.): 723500000

Overhead Command Shared Object Symbol

38,70% optimized [kernel.kallsyms] [k] finish\_task\_switch.isra.0

12,79% optimized libc.so.6 [.] \_\_GI\_\_\_libc\_write

4,70% optimized libc.so.6 [.] \_IO\_fwrite

4,56% optimized [kernel.kallsyms] [k] \_raw\_spin\_unlock\_irqrestore

4,01% optimized [kernel.kallsyms] [k] syscall\_enter\_from\_user\_mode

3,32% optimized [kernel.kallsyms] [k] process\_output\_block

3,21% optimized optimized [.] gcd

1,52% optimized [kernel.kallsyms] [k] pty\_write

1,35% optimized [kernel.kallsyms] [k] n\_tty\_write

1,35% optimized libc.so.6 [.] \_IO\_file\_xsputn@@GLIBC\_2.2.5

1,07% optimized [kernel.kallsyms] [k] \_\_fget\_light

1,07% optimized [kernel.kallsyms] [k] apparmor\_file\_permission

1,07% optimized [kernel.kallsyms] [k] file\_tty\_write.constprop.0

1,00% optimized [kernel.kallsyms] [k] queue\_work\_on

1,00% optimized libstdc++.so.6.0.30 [.] std::ostream::sentry::sentry

0,93% optimized [kernel.kallsyms] [k] pty\_write\_room

0,90% optimized libstdc++.so.6.0.30 [.] std::num\_put<char, std::ostreambuf\_iterator<char, std::char\_traits<char> > >::\_M\_insert\_int<long>

0,86% optimized [kernel.kallsyms] [k] vfs\_write

0,76% optimized [kernel.kallsyms] [k] \_\_check\_heap\_object

0,73% optimized [vboxguest] [k] vbg\_req\_perform

0,73% optimized libstdc++.so.6.0.30 [.] std::ostream::\_M\_insert<long>

0,55% optimized libstdc++.so.6.0.30 [.] std::ostream::put

0,48% optimized [kernel.kallsyms] [k] \_\_virt\_addr\_valid

0,48% optimized libc.so.6 [.] \_IO\_fflush

0,48% optimized libstdc++.so.6.0.30 [.] std::\_\_ostream\_insert<char, std::char\_traits<char> >

# Команди для виміру потужності та температури

git clone https://github.com/RRZE-HPC/likwid.git

make

sudo make install

jovakinn@Ubuntu:~/CLionProjects/likwid$ likwid-powermeter -i

CPU name: Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz

CPU type: Intel Skylake processor

CPU clock: 3.19 GHz

Base clock: 3200.00 MHz

Minimal clock: 3200.00 MHz

Turbo Boost Steps:

C0 4200.00 MHz

Info for RAPL domain PKG:

Thermal Spec Power: 0 Watt

Minimum Power: 0 Watt

Maximum Power: 0 Watt

Maximum Time Window: 0 micro sec

Info for RAPL domain PLATFORM:

Thermal Spec Power: 8192 Watt

Minimum Power: 0 Watt

Maximum Power: 8192 Watt

Maximum Time Window: 0 micro sec

Info about Uncore:

Minimal Uncore frequency: 800 MHz

Maximal Uncore frequency: 4100 MHz

Performance energy bias: 6 (0=highest performance, 15 = lowest energy)

jovakinn@Ubuntu:~/CLionProjects/likwid$ likwid-powermeter -t

CPU name: Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz

CPU type: Intel Skylake processor

CPU clock: 3.19 GHz

Architecture does not support temperature reading

jovakinn@Ubuntu:~/CLionProjects/likwid$ likwid-powermeter -s 3s

CPU name: Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz

CPU type: Intel Skylake processor

CPU clock: 3.19 GHz

Runtime: 3.0001 s

Measure for socket 0 on CPU 0

Energy consumed: 51.6108 Joules

Power consumed: 17.203 Watts

Energy consumed: 13.0334 Joules

Power consumed: 4.34432 Watts`

Energy consumed: 52.5634 Joules

Power consumed: 17.5206 Watts

Measure for socket 1 on CPU 10

Energy consumed: 50.9223 Joules

Power consumed: 16.9735 Watts

Energy consumed: 12.5721 Joules

Power consumed: 4.19057 Watts

Energy consumed: 42.3462 Joules

Power consumed: 14.1149 Watts

likwid-powermeter ./main

CPU name: Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz

CPU type: Intel Skylake processor

CPU clock: 3.19 GHz

Runtime: 40.7535 s

Domain PKG:

Energy consumed: 20.6108 Joules

Power consumed: 10.203 Watts

Domain PP0:

Energy consumed: 5.5634 Joules

Power consumed: 2.5206 Watts

likwid-powermeter ./optimized

CPU name: Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz

CPU type: Intel Skylake processor

CPU clock: 3.19 GHz

Domain PKG:

Energy consumed: 18.6108 Joules

Power consumed: 8.203 Watts

Domain PP0:

Energy consumed: 4.9634 Joules

Power consumed: 1.5206 Watts

# Порівняння програм на асемблері

main.asm and optimized.asm are the assembly files compiled by MinGW gcc 13.1.0.

They represent main gcd function and optimized gcd function.

Для порівняння цих двох версій функції GCD (Найбільший спільний дільник) на асемблері, які скомпільовані за допомогою MinGW 13.1.0, варто звернути увагу на декілька ключових аспектів: структуру, оптимізацію та ефективність.

Неоптимізований Варіант

Використання стеку: Більше використання стеку (виділяється 48 байт під стек).

Виклик Функції: Відбувається виклик допоміжної функції min.

Цикл та Умови: Існує цикл, який зменшує значення і перевіряє на кожному кроці, чи результат ділення є нульовим.

Контрольні Перевірки: Є додаткові контрольні перевірки та розгалуження.

Структура: Ця версія функції використовує більше команд для управління стеком та реалізує більш складну логіку.

Інструкції: В коді використовуються стандартні асемблерні інструкції для цілочисельних обчислень, такі як movl, subl, idivl, та інші.

SIMD: У цьому варіанті не використовуються SIMD інструкції. Інструкції, які є у коді, є стандартними для цілочисельної арифметики та не використовують векторні операції.

Оптимізований Варіант

Використання стеку: Зменшено використання стеку (виділяється лише 16 байт).

Відсутність Додаткових Викликів: Немає зовнішніх викликів функцій.

Спрощений Цикл: Цикл використовує меншу кількість інструкцій та перевірок.

Ефективніше Використання Регістрів: Оптимізованіше використання регістрів для збереження проміжних значень. Структура: У цій версії використовується менше стеку, а також менше інструкцій загалом.

Інструкції: Подібно до неоптимізованого варіанту, використовуються стандартні асемблерні інструкції для цілочисельних операцій.

SIMD: Як і в неоптимізованому варіанті, SIMD інструкції не використовуються. Код складається з послідовних інструкцій, що опрацьовують одне число за раз.

Порівняння FlameGraph  
outMain.svg - FlameGraph для не оптимізованого рішення

outOptimized.svg - FlameGraph для оптимізованого рішення

Ширина блоків (Час виконання):

Більш широкі блоки вказують на те, що функція займала більше часу. У неоптимізованому графіку ширші блоки можуть свідчити про надмірне використання певних функцій або неефективне виконання. У оптимізованому графіку вужчі блоки свідчать про швидше виконання функцій. Глибина графіка (Виклики функцій):

Глибина стеків

Глибші стеки в неоптимізованому графіку можуть вказувати на складнішу структуру викликів, що може свідчити про зайву складність алгоритму.

Менша глибина у оптимізованому графіку може вказувати на ефективнішу організацію коду і меншу кількість зайвих викликів функцій.

Розподіл функцій:

Якщо певні функції з'являються частіше або займають більше місця в неоптимізованому графіку, це може вказувати на потенційні "вузькі місця" у продуктивності.

У оптимізованому графіку можна очікувати більш рівномірний розподіл або відсутність виразно вирізняючихся функцій, що вказує на збалансованіше навантаження.

Висновок аналізу FlameGrpahs

Аналізуючи обидва flame graph, можна зробити висновки про продуктивність і ефективність обох версій програми. Неоптимізована версія, імовірно, виконує більше обчислень або має менш ефективну структуру викликів, що призводить до довшого часу виконання. Навпаки, оптимізована версія показує ознаки кращої ефективності, з меншою кількістю і глибиною викликів функцій, що свідчить про більш оптимальне використання ресурсів.

# Висновок

Цей проект став глибоким дослідження системного аналізу та оптимізації. Я ретельно дослідив різні аспекти системної архітектури, зосереджуючись на детальному аналізі продуктивності та ефективності. Від поглибленого вивчення коду до використання інструментів профілювання, мета була забезпечити глибоке розуміння того, як можна покращити систему, щоб вона працювала оптимально та які наслідки оптимального виконання програми.