Київський національний університет імені Тараса Шевченка
факультет радіофізики, електроніки та комп'ютерних систем

Лабораторна робота № 3

Тема: «Дослідження оптимізації коду з використанням векторних розширень CPU**»**

Роботу виконав
студент 3 курсу
КІ - СА
Гулівець Владислав
Андрійович

Київ 2020

Хід роботи

1. Отримайте доступ на обчислювальний кластер для роботи з Intel Compiler.

```
Last login: Mon Mar 30 22:46:58 EEST 2020 on pts/2
[tb287@plus7 ~]$
```

2. Завантажте файли Intel® C++ Compiler - Using Auto-Vectorization Tutorial (https://software.intel.com/en-us/product-code-samples?topic=20813) на свій комп'ютер та в домашню директорію користувача обчислювального кластеру.

```
[tb287@plus7 ~]$ ls
advisor          compiler_c  example   index.html  intel  ITAC       mkl   tbb
cluster_checker  compiler_f  example2  inspector   ipp    licensing  pstl  vtune_amplifier
```

3. Використовуючи інструкції в readme.html ознайомтесь та виконайте Tutorial на обчислювальному кластері

*Замість інструкцій в пункті "Setting the Environment Variables" завантажте оточення компілятора шляхом виконання команди: `ml icc`

*Виконуйте завдання на робочих вузлах кластеру замість вхідної ноди. По-перше процесори робочих вузлів мають набагато більше розширень. По-друге виконання компіляції та запуску на вхідній ноді заважає іншим користувачам, що призведе до **блокування вашого аккаунту та автоматичного незарахування лабораторної роботи**. Рекомендований варіант виконання роботи - використання інтерактивних задач в системі планування:

```
[manf@plus7 ~]$ qsub -I -l nodes=1:ppn=1,walltime=00:30:00
KNU:WN:s5 [manf ~]$ ml icc
```

Generating a Vectorization Report

```
Intel(R) Advisor can now assist with vectorization and show optimization
  report messages with your source code.
See "https://software.intel.com/en-us/intel-advisor-xe" for details.


Begin optimization report for: matvec(int, int, double (*)[*], double *, double *)

    Report from: Vector optimizations [vec]


LOOP BEGIN at src/Multiply.c(37,5)
   remark #25460: No loop optimizations reported

   LOOP BEGIN at src/Multiply.c(49,9)
      remark #25460: No loop optimizations reported
   LOOP END

   LOOP BEGIN at src/Multiply.c(49,9)
   <Remainder>
   LOOP END
LOOP END
```

```
ROW:101 COL: 101
Execution time is 6.729 seconds
GigaFlops = 3.031770
Sum of result = 195853.999899
```

За аналогією були проведені наступні команди в Tutorial-list.

4. Оберіть будь-яку неінтреактивну консольну програму мовою C/C++ (унікальну в межах групи, в гуглі більше ніж 50 програм)

*Напишіть сценарій, що:
*Компілює програму з різними оптимізаціями (-O) та виміряйте час її роботи. Якщо час досить малий - вимірюйте час роботи 1000 (чи 1000000) запусків алгоритму в циклі. Час роботи можна виміряти утилітою time.

*Отримує перелік всіх розширень процесору що підтримуються
*Для кожного розширення компілює Intel-компілятором окремий варіант оптимізованого коду (наприклад -x SSE2)
*Вимірює час виконання кожного варіанта оптимізованої програми

*Запустіть задачу **в планувальник** обчислювального кластеру 5 разів (для статистики на різних нодах)
[manf@plus7 ~]$ qsub -N MyJob -l nodes=1:ppn=1,walltime=00:30:00 script.sh

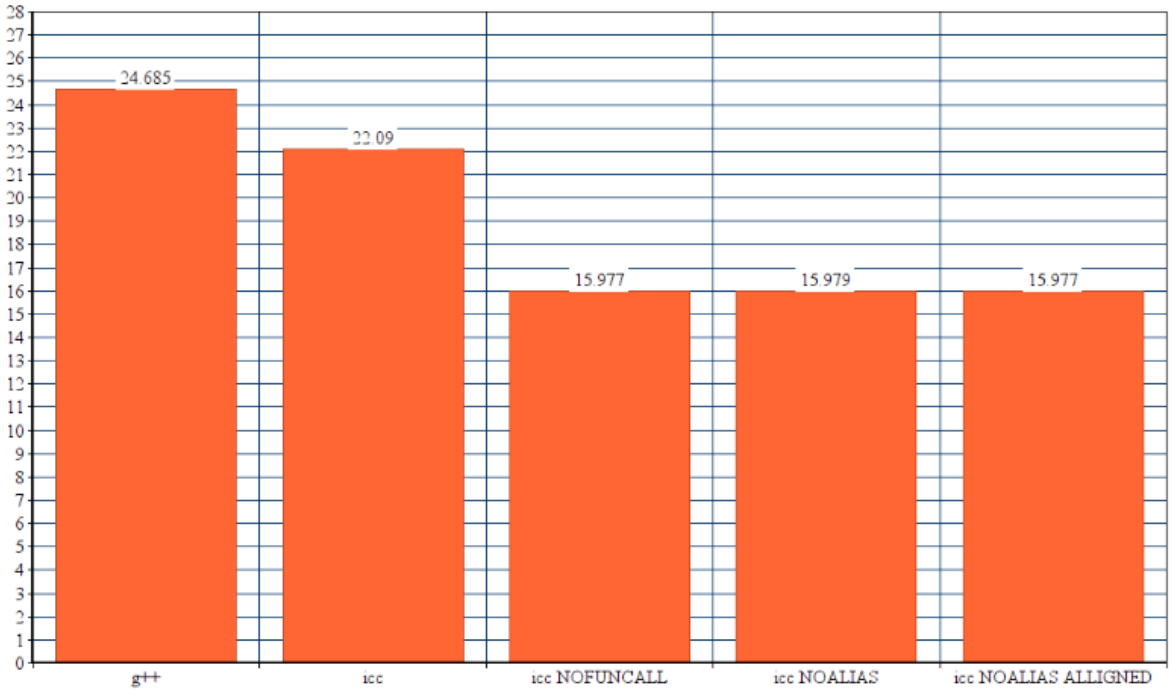
Побудуйте графіки залежності часу від різних варіантів компіляції.

```cpp
#include <iostream>
using namespace std;
int Fib(int i)
{
int value = 0;
if(i < 1) return 0;
if(i == 1) return 1;
return Fib(i-1) + Fib(i - 2);
}
int main()
{
int i = 0;
while(i < 45 )
{ cout << Fib(i) << endl;
i++;
}
return 0;
}
```

Код моєї неінтерактивної програми

```
real    0m24.685s
user    0m24.679s
sys     0m0.003s

real    0m22.090s
user    0m22.085s
sys     0m0.002s
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location

real    0m15.977s
user    0m15.968s
sys     0m0.006s
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location

real    0m15.979s
user    0m15.972s
sys     0m0.004s
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location

real    0m15.977s
user    0m15.970s
sys     0m0.004s
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location

real    0m15.977s
user    0m15.972s
sys     0m0.002s
```

Результат виконання скрипта



Часові інтервали виконання програми

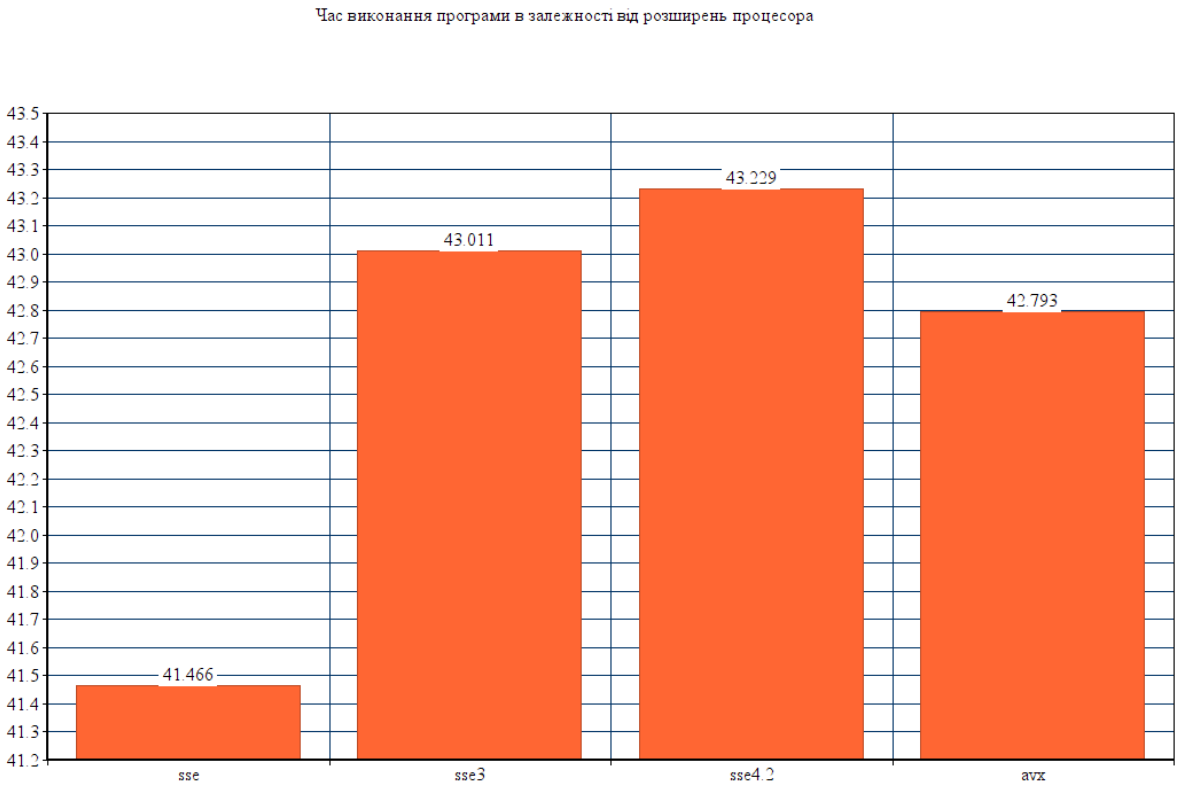
```
cd /home/grid/testbed/tb287/example
ml icc
g++ fibonacci.cpp -o sample0
time ./sample0
icc -O1 fibonacci.cpp -o sample1
time ./sample1
icc -O2 -D NOFUNCCALL -qopt-report=2 -qopt-report-phase=vec fibonacci.cpp -o sample2
time ./sample2
icc -qopt-report=2 -qopt-report-phase=vec -D NOALIAS fibonacci.cpp -o sample3
time ./sample3
icc -qopt-report=4 -qopt-report-phase=vec -D NOALIAS -D ALIGNED fibonacci.cpp -o sample4
time ./sample4
icc -qopt-report=2 -qopt-report-phase=vec -D NOALIAS -D ALIGNED -ipo fibonacci.cpp -o sample5
time ./sample5
```

Сам скрипт

```
[tb287@plus7 example2]$ grep "flags" /proc/cpuinfo | uniq
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ss syscall nx pd
pe1gb rdtscp lm constant_tsc arch_perfmon rep_good nopl xtopology eagerfpu pni pclmulqdq ssse3 cx16 pcid sse4_1 sse4_2 x2apic popcnt
 tsc_deadline_timer aes xsave avx hypervisor lahf_lm tsc_adjust xsaveopt ibpb ibrs stibp arat spec_ctrl intel_stibp
```

Перелік розширень процесора

Для того щоб перегляду усіх доступних флагів було використано команду, а потім вибрано флаги, які відповідають розширенням.
grep “flags” /proc/cpuinfo | uniq



Час виконання програми в залежності від розширень процесора

```
#!/bin/bash
ml icc

extens='sse2 ssse3 ss4.1 sse4.2 avx'
for i in $extens
do
icc -std=c++11 -x$i -O2 fibonacci.cpp -o resextn
if [ $? -eq 0 ]
then
echo""
echo "Extension is $i"
time ./resextn
fi
done
```

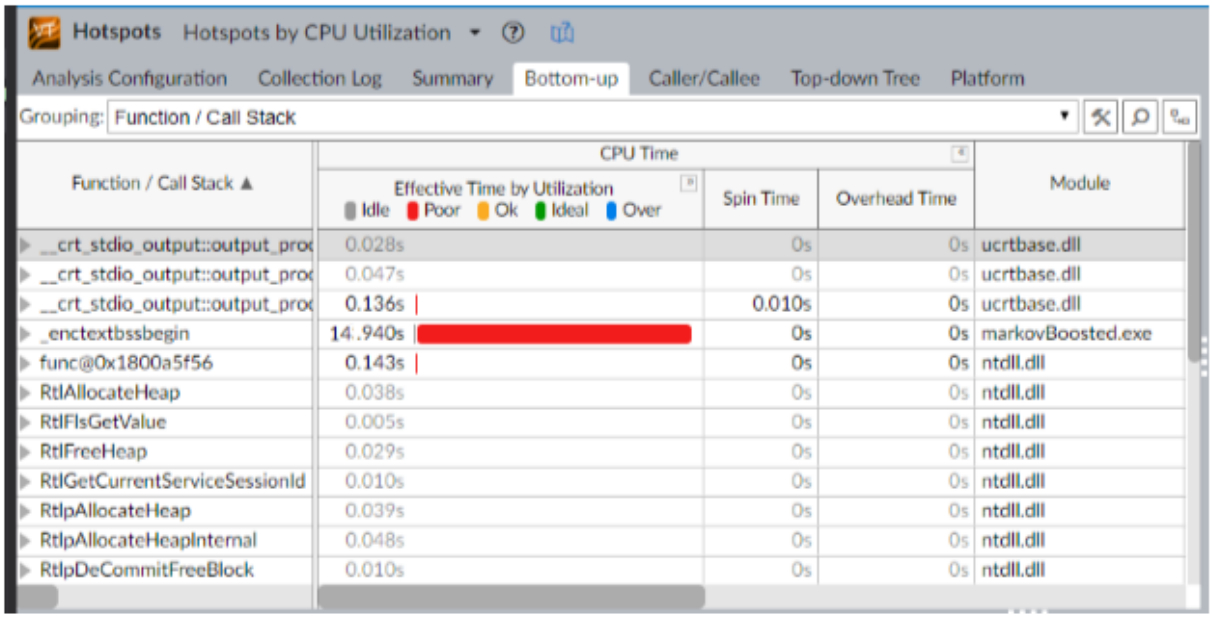
Сам скрипт що застосовує розширення процесора

```
[tb287@plus7 example]$ ls
fibonacci.cpp     ipo_out.optrpt  node2.o2642037  node3.o2642039  node4.o2642041  node5.o2642042
fibonacci.optrpt  node2.e2642037  node3.e2642039  node4.e2642041  node5.e2642042  sample0
```

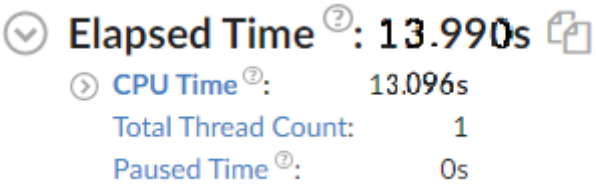
Виконання програми на 5 різних нодах

Оберіть будь-який зі створених вами програмних продуктів та виконайте його оптимізацію з використання Intel® Parallel Studio.

Час без оптимізації:



Час зі застосунком оптимізації:



Висновок: У ході виконання лабораторної роботи мною було досліджено та проаналізовано способи компіляції та оптимізації програми за допомогою різних векторних розширень процесора. Мною було розібрана основа роботи на обчислювальному кластері університету, робота з Intel C++ Compiler. Мною було побудовані часові діаграми, що допомагають виявити різницю у часі компіляції\виконання відповідно до розширення та рівня оптимізації.

My personal GitHub link: ***https://github.com/gulivec84***