Київський національний університет імені Тараса Шевченка
радіофізичний факультет

Лабораторна робота № 3

Тема: Дослідження оптимізації коду з використанням векторних розширень CPU

Роботу виконав
студент 3 курсу
Комп’ютерної інженерії
Веремій Юрій

Київ 2019

Силка на git https://github.com/uayura/koputer_sistem/tree/master/lab3

1. **Отримайте доступ на обчислювальний кластер для роботи з Intel Compiler**
2. **Завантажте файли Intel® C++ Compiler - Using Auto-Vectorization Tutorial.**

```
[tb445@plus7 ~]$ ls
vec_samples  vec_samples_C_lin_20170911.tgz
[tb445@plus7 ~]$
```

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

```
KNU:  :s1 [tb445 src]$ icc -O1 -std=c99 Multiply.c Driver.c -o MatVector
KNU:  :s1 [tb445 src]$./MatVector

ROW:101 COL: 101
Execution time is 12.051 seconds
GigaFlops = 1.692943
Sum of result = 195853.999899
KNU:  :s1 [tb445 src]$
```

```
[tb445@plus7 ~]$ qsub -I -l nodes=1:ppn=1,walltime=00:30:00
qsub: waiting for job 2789310 to start
qsub: job 2789310 ready

autoscratch: creating directory '/mnt/work/tb445'
autoscratch: creating directory '/mnt/scratch/tb445'
KNU:  :s1 [tb445 ~]$ ml icc
KNU:  :s1 [tb445 ~]$
```

```
KNU:  :s1 [tb445 src]$ icc -std=c99 -O2 -D NOFUNCCALL -qopt-report=1 -qopt-report-phase=vec Multiply.c D
river.c -o MatVector
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location
KNU:  :s1 [tb445 src]$ cat Multiply.optrpt
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 Multiply.c(37,5)
   remark #25460: No loop optimizations reported

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

   LOOP BEGIN at Multiply.c(49,9)
   <Remainder>
   LOOP END
LOOP END
===========================================================================
KNU:  :s1 [tb445 src]$ ./MatVector

ROW:101 COL: 101
Execution time is 4.103 seconds
GigaFlops = 4.972197
Sum of result = 195853.999899
KNU:  :s1 [tb445 src]$
```

```
KNU:  :s1 [tb445 src]$ icc -std=c99 -O2 -D NOFUNCCALL -qopt-report-phase=vec,loop -qopt-report=2 Multipl
y.c Driver.c -o MatVector
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location
KNU:  :s1 [tb445 src]$ cat Multiply.optrpt
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: Loop nest & Vector optimizations [loop, vec]


LOOP BEGIN at Multiply.c(37,5)
   remark #15541: outer loop was not auto-vectorized: consider using SIMD directive

   LOOP BEGIN at Multiply.c(49,9)
      remark #15344: loop was not vectorized: vector dependence prevents vectorization. First dependence
 is shown below. Use level 5 report for details
      remark #15346: vector dependence: assumed FLOW dependence between b[i] (50:13) and b[i] (50:13)
      remark #25439: unrolled with remainder by 2
   LOOP END

   LOOP BEGIN at Multiply.c(49,9)
   <Remainder>
   LOOP END
LOOP END
===========================================================================
KNU:  :s1 [tb445 src]$ ./MatVector

ROW:101 COL: 101
Execution time is 4.098 seconds
GigaFlops = 4.978424
Sum of result = 195853.999899
KNU:  :s1 [tb445 src]$
```

```
KNU:  :s1 [tb445 src]$ icc -std=c99 -qopt-report=2 -qopt-report-phase=vec -D NOALIAS Multiply.c Driver.c
 -o MatVector
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location
KNU:  :s1 [tb445 src]$ cat Multiply.optrpt
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 *__restrict__, double *)

    Report from: Vector optimizations [vec]


LOOP BEGIN at Multiply.c(37,5)
   remark #15542: loop was not vectorized: inner loop was already vectorized

   LOOP BEGIN at Multiply.c(49,9)
   <Peeled loop for vectorization>
   LOOP END

   LOOP BEGIN at Multiply.c(49,9)
      remark #15300: LOOP WAS VECTORIZED
   LOOP END

   LOOP BEGIN at Multiply.c(49,9)
   <Alternate Alignment Vectorized Loop>
   LOOP END

   LOOP BEGIN at Multiply.c(49,9)
   <Remainder loop for vectorization>
   LOOP END
LOOP END
================================================================================
KNU:  :s1 [tb445 src]$ ./MatVector

ROW:101 COL: 101
Execution time is 4.599 seconds
GigaFlops = 4.436404
Sum of result = 195853.999899
KNU:  :s1 [tb445 src]$
```

```
KNU:  :s1 [tb445 src]$ icc -std=c99 -qopt-report=4 -qopt-report-phase=vec -D NOALIAS -D ALIGNED Multiply
.c Driver.c -o MatVector
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location
KNU:  :s1 [tb445 src]$ cat Multiply.optrpt
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.

Intel(R) C Intel(R) 64 Compiler for applications running on Intel(R) 64, Version 18.0.5.274 Build 201808
23

Compiler options: -std=c99 -qopt-report=4 -qopt-report-phase=vec -D NOALIAS -D ALIGNED -o MatVector

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

    Report from: Vector optimizations [vec]


LOOP BEGIN at Multiply.c(37,5)
   remark #15542: loop was not vectorized: inner loop was already vectorized

   LOOP BEGIN at Multiply.c(49,9)
      remark #15388: vectorization support: reference a[i][j] has aligned access   [ Multiply.c(50,21) ]
      remark #15388: vectorization support: reference x[j] has aligned access   [ Multiply.c(50,31) ]
      remark #15305: vectorization support: vector length 2
      remark #15399: vectorization support: unroll factor set to 4
      remark #15309: vectorization support: normalized vectorization overhead 0.594
      remark #15300: LOOP WAS VECTORIZED
      remark #15448: unmasked aligned unit stride loads: 2
      remark #15475: --- begin vector cost summary ---
      remark #15476: scalar cost: 10
      remark #15477: vector cost: 4.000
      remark #15478: estimated potential speedup: 2.410
      remark #15488: --- end vector cost summary ---
   LOOP END

   LOOP BEGIN at Multiply.c(49,9)
   <Remainder loop for vectorization>
      remark #15388: vectorization support: reference a[i][j] has aligned access   [ Multiply.c(50,21) ]
      remark #15388: vectorization support: reference x[j] has aligned access   [ Multiply.c(50,31) ]
      remark #15335: remainder loop was not vectorized: vectorization possible but seems inefficient. Us
e vector always directive or -vec-threshold0 to override
      remark #15305: vectorization support: vector length 2
      remark #15309: vectorization support: normalized vectorization overhead 2.417
   LOOP END
LOOP END
================================================================================
KNU:  :s1 [tb445 src]$
```

```
   LOOP END
LOOP END
================================================================================
KNU:  :s1 [tb445 src]$ ./MatVector

ROW:101 COL: 102
Execution time is 4.251 seconds
GigaFlops = 4.799581
Sum of result = 195853.999899
KNU:  :s1 [tb445 src]$
```

```
KNU:  :s1 [tb445 src]$
KNU:  :s1 [tb445 src]$ icc -std=c99 -qopt-report=2 -qopt-report-phase=vec -D NOALIAS -D ALIGNED -ipo Mul
tiply.c Driver.c -o MatVector
icc: remark #10397: optimization reports are generated in *.optrpt files in the output location
KNU:  :s1 [tb445 src]$ ls
Driver.c  Driver.optrpt  MatVector  Multiply.c  Multiply.h  Multiply.optrpt  ipo_out.optrpt
KNU:  :s1 [tb445 src]$ cat ipo_out.optrpt
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: main()

    Report from: Vector optimizations [vec]


LOOP BEGIN at Driver.c(152,16)
   remark #15542: loop was not vectorized: inner loop was already vectorized

   LOOP BEGIN at Multiply.c(37,5) inlined into Driver.c(150,9)
      remark #15542: loop was not vectorized: inner loop was already vectorized

      LOOP BEGIN at Multiply.c(49,9) inlined into Driver.c(150,9)
         remark #15300: LOOP WAS VECTORIZED
      LOOP END

      LOOP BEGIN at Multiply.c(49,9) inlined into Driver.c(150,9)
      <Remainder loop for vectorization>
         remark #15335: remainder loop was not vectorized: vectorization possible but seems inefficient.
 Use vector always directive or -vec-threshold0 to override
      LOOP END
   LOOP END
LOOP END

LOOP BEGIN at Driver.c(74,5) inlined into Driver.c(159,5)
   remark #15300: LOOP WAS VECTORIZED
LOOP END

LOOP BEGIN at Driver.c(74,5) inlined into Driver.c(159,5)
<Remainder loop for vectorization>
LOOP END
===========================================================================

Begin optimization report for: init_matrix(int, int, double, double (*)[102])

    Report from: Vector optimizations [vec]


LOOP BEGIN at Driver.c(47,5)
   remark #15542: loop was not vectorized: inner loop was already vectorized
```

```
<Remainder loop for vectorization>
LOOP END
===========================================================================

Begin optimization report for: init_matrix(int, int, double, double (*)[102])

    Report from: Vector optimizations [vec]


LOOP BEGIN at Driver.c(47,5)
   remark #15542: loop was not vectorized: inner loop was already vectorized

   LOOP BEGIN at Driver.c(48,9)
      remark #15300: LOOP WAS VECTORIZED
   LOOP END

   LOOP BEGIN at Driver.c(48,9)
   <Remainder loop for vectorization>
   LOOP END
LOOP END

LOOP BEGIN at Driver.c(53,9)
   remark #15300: LOOP WAS VECTORIZED
LOOP END

LOOP BEGIN at Driver.c(53,9)
<Remainder loop for vectorization>
LOOP END
===========================================================================

Begin optimization report for: init_array(int, double, double *)

    Report from: Vector optimizations [vec]


LOOP BEGIN at Driver.c(62,5)
   remark #15300: LOOP WAS VECTORIZED
LOOP END

LOOP BEGIN at Driver.c(62,5)
<Remainder loop for vectorization>
LOOP END
===========================================================================
KNU:  :s1 [tb445 src]$ ./MatVector

ROW:101 COL: 102
Execution time is 3.985 seconds
GigaFlops = 5.119741
Sum of result = 195853.999899
KNU:  :s1 [tb445 src]$
```

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

Branch: master ▾ koputer_sistem / lab3 / Sieve_of_Eratosthenes.cpp Find file Copy path

uayura Update Sieve_of_Eratosthenes.cpp 025c7b1 a minute ago

1 contributor

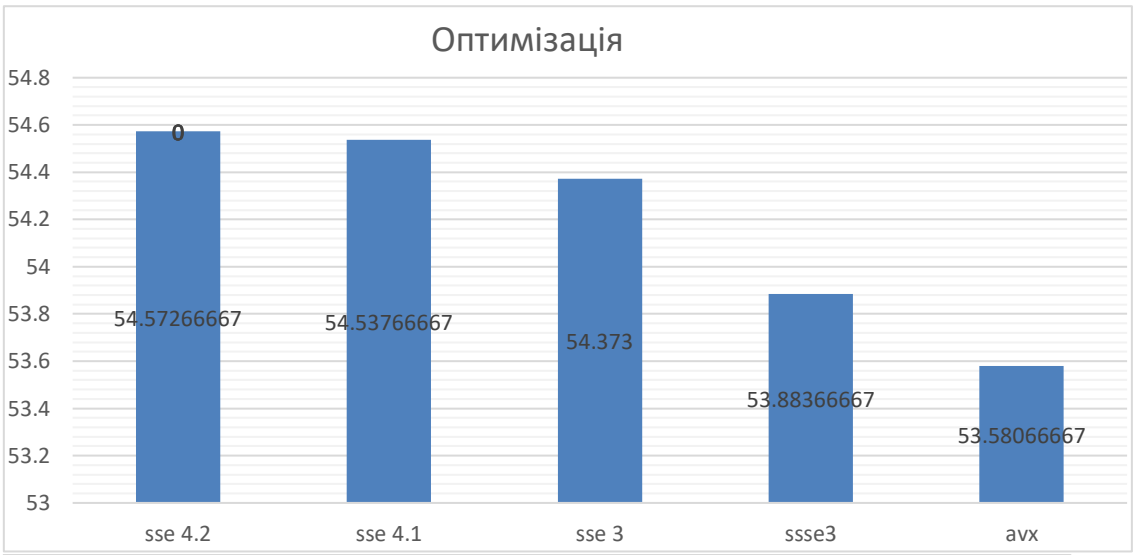
26 lines (24 sloc) 463 Bytes Raw Blame History

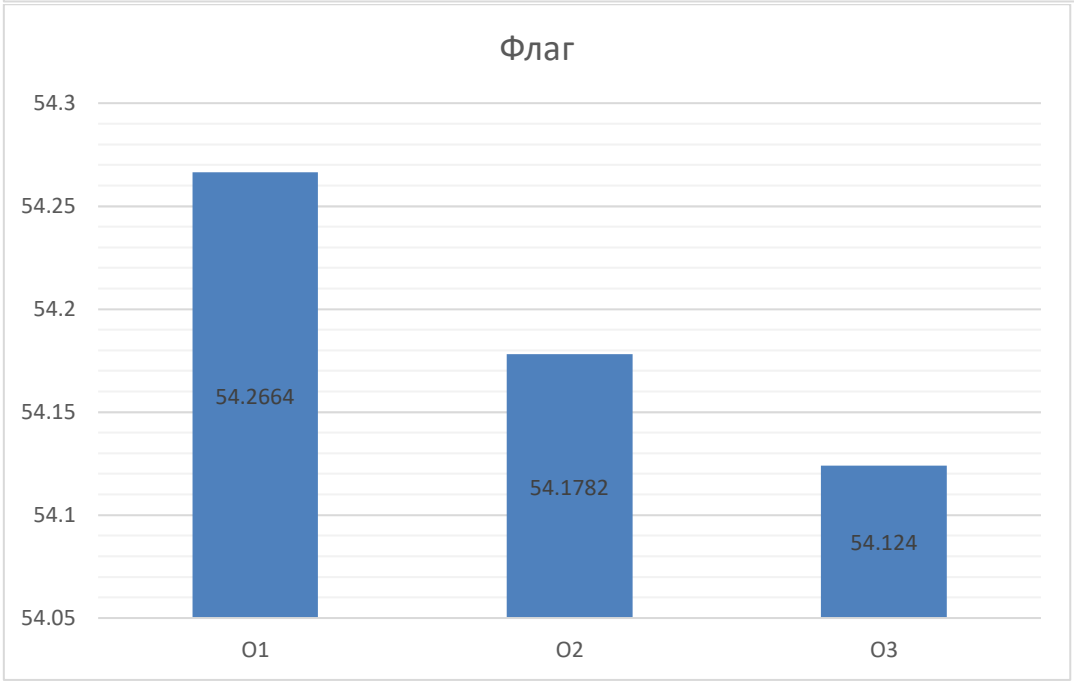
```cpp
//https://prog-cpp.ru/eratosfen/
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
	long long n;
	n = 2000000;
	long long *a = new long long[n + 1];
	for (long long i = 0; i < n + 1; i++)
		a[i] = i;
	ofstream fout;
	fout.open("file.txt");
	for (long long p = 2; p < n + 1; p++)
	{
		if (a[p] != 0)
		{
			fout << a[p] << endl;
			for (long long j = p * p; j < n + 1; j += p)
				a[j] = 0;
		}
	}
	fout.close();
}
```

```
#!/bin/bash
flags=( "sse4.2" "sse4.1" "sse3" "ssse3" "avx" )
for i in "${flags[@]}";do
	for j in {1..3};do
		icc -O$j -m$i Sieve_of_Eratosthenes.cpp -o temp
		echo
		echo $i "  -  " $j
		time `for i in {0..50}; do ./temp; done`
	done
done

~
```

	sse 4.2	sse 4.1	sse 3	ssse3	avx	Avg
O1	54.173	55.825	53.83	54.199	53.305	**54.2664**
O2	54.137	53.728	55.982	53.9	53.144	**54.1782**
O3	55.408	54.06	53.307	53.552	54.293	**54.124**
Avg	**54.57267**	**54.53767**	**54.373**	**53.88367**	**53.58067**	





Як бачимо з отриманих даних, найшвидше програма виконується у випадку компіляції з третім методом оптимізації та розширенням процесора avx.

```
[tb445@plus7 ~]$ ls
file.txt  lab3.sh  Sieve_of_Eratosthenes.cpp  temp  test  vec_samples  vec_samples_C_lin_20170911.tgz
[tb445@plus7 ~]$ qsub -N MyJob -l nodes=1:ppn=1,walltime=00:30:00 lab3.sh
2789450
[tb445@plus7 ~]$ qsub -N MyJob -l nodes=2:ppn=1,walltime=00:30:00 lab3.sh
2789452
[tb445@plus7 ~]$ qsub -N MyJob -l nodes=3:ppn=1,walltime=00:30:00 lab3.sh
2789453
[tb445@plus7 ~]$ qsub -N MyJob -l nodes=4:ppn=1,walltime=00:30:00 lab3.sh
2789455
[tb445@plus7 ~]$ qsub -N MyJob -l nodes=5:ppn=1,walltime=00:30:00 lab3.sh
2789456
[tb445@plus7 ~]$
```

5. Виконання всіх попередніх пунктів оцінюється в 8 балів. Для отримання 10 балів виконайте наступне:

це для Visual C++:

Elapsed Time: 3.608s

CPU Time: 3.124s

Total Thread Count: 1

Paused Time: 0s

Hotspots Insights

If you see significant hotspots in the Top Hotspots list, switch to the Bottom-up view for in-depth analysis per function. Otherwise, use the Caller/Callee view to track critical paths for these hotspots.

Top Hotspots

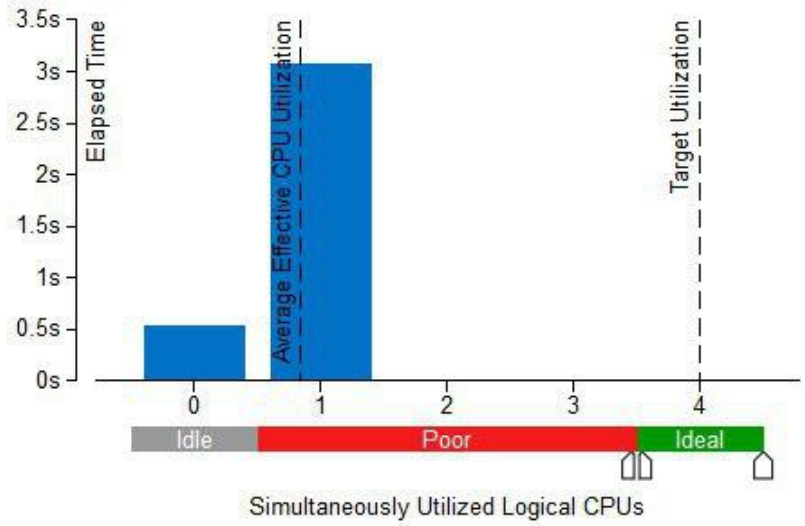
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
std::basic_istream<char,struct std::char_traits<char> >::operator>>	msvcp140.dll	1.996s
std::basic_ostream<char,struct std::char_traits<char> >::operator<<	msvcp140.dll	0.896s
std::endl<char,struct std::char_traits<char> >	BTP.exe	0.090s
main	BTP.exe	0.060s
[Outside any known module]		0.022s
[Others]		0.060s

**N/A is applied to non-summable metrics.*

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.



це для Intel C++:

Elapsed Time: 3.452s

CPU Time:	3.065s
Total Thread Count:	1
Paused Time:	0s

Hotspots Insights

If you see significant hotspots in the Top Hotspots list, switch to the Bottom-up view for in-depth analysis per function. Otherwise, use the Caller/Callee view to track critical paths for these hotspots.

Top Hotspots

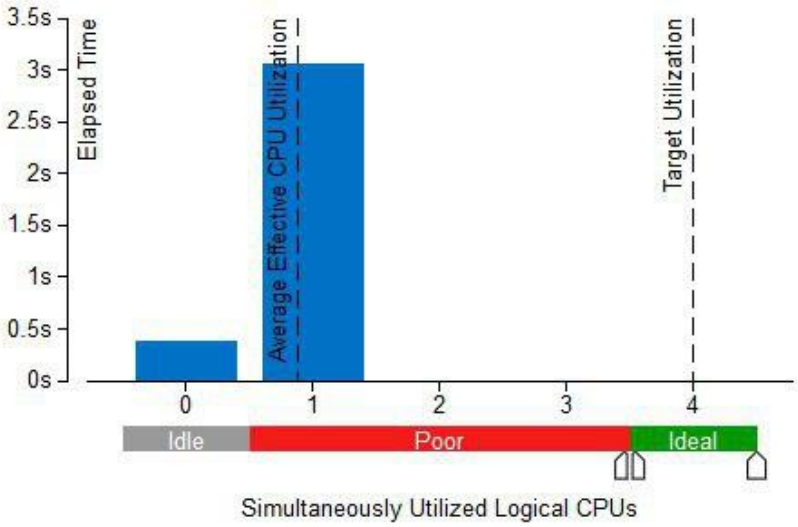
This section lists the most active functions in your application. Optimizing these hotspot functions typically results in improving overall application performance.

Function	Module	CPU Time
std::basic_istream <char,struct std::char_traits<char> >:: operator>>	msvcp140.dll	1.972s
std::basic_ostream <char,struct std::char_traits<char> >:: operator<<	msvcp140.dll	0.856s
std::endl<char,struct std::char_traits< char> >	BTP.exe	0.120s
main	BTP.exe	0.073s
exit	ucrtbase.dll	0.015s
[Others]		0.030s

**N/A is applied to non-summable metrics.*

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.



Висновок: У результаті виконання даної лабораторної роботи було проведено ознайомлення з обчислювальним кластером та методами оптимізації виконання коду процесором.