

# SIMD demo

## 1) Set number of threads:

```
export OMP_NUM_THREADS=4
```

## 2) Set CPU affinity:

```
export GOMP_CPU_AFFINITY=1,2,3,4
```

## 3) Check:

```
cpuinfo
```

here is the output:

```
simd> cpuinfo
```

```
Intel(R) processor family information utility, Version 2021.1-  
beta10 Build 20201007 (id: 81f9c8f47)  
Copyright (C) 2005-2020 Intel Corporation. All rights reserved.
```

```
==== Processor composition ====
```

```
Processor name      : Intel(R) Core(TM) i7-7700HQ  
Packages(sockets)  : 1  
Cores               : 4  
Processors(CPUs)   : 8  
Cores per package  : 4  
Threads per core   : 2
```

```
==== Processor identification ====
```

Processor	Thread Id.	Core Id.	Package Id.
0	0	0	0
1	0	1	0
2	0	2	0
3	0	3	0
4	1	0	0
5	1	1	0
6	1	2	0
7	1	3	0

```
==== Placement on packages ====
```

Package Id.	Core Id.	Processors
0	0,1,2,3	(0,4)(1,5)(2,6)(3,7)

```
==== Cache sharing ====
```

Cache	Size	Processors
L1	32 KB	(0,4)(1,5)(2,6)(3,7)
L2	256 KB	(0,4)(1,5)(2,6)(3,7)
L3	6 MB	(0,1,2,3,4,5,6,7)

```
simd>
```

#### 4) Prepare the compilation environment

```
simd> source ~/science/opt/intel/oneapi/setvars.sh
:: initializing oneAPI environment ...
   BASH version = 5.0.17(1)-release
:: vpl -- latest
:: mkl -- latest
:: inspector -- latest
:: dpcpp-ct -- latest
:: vtune -- latest
:: debugger -- latest
:: mpi -- latest
:: clck -- latest
:: dpl -- latest
:: intelpython -- latest
:: itac -- latest
:: ipp -- latest
:: ippcp -- latest
:: tbb -- latest
:: compiler -- latest
:: dev-utilities -- latest
:: ccl -- latest
:: dal -- latest
:: dnnl -- latest
:: advisor -- latest
:: oneAPI environment initialized ::
```

#### 5) Compile:

```
simd> icc -O3 -qopt-report -qopenmp -o ./simd_example
./simd_example.c
icc: remark #10397: optimization reports are generated in *.optrpt
files in the output location
```

#### 6) Execute:

```
./simd_example
```

```
simd> ./simd_example
serial computation
compute time=0.007758
```

```
Parallel loop
compute time=0.009401
```

```
Parallel loop+simd
compute time=0.008552
```

#### 7) Check the optimization report

```
simd> code ./simd_example.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.

Report from: Interprocedural optimizations [ipo]

INLINING OPTION VALUES:

- inline-factor: 100
- inline-min-size: 30
- inline-max-size: 230
- inline-max-total-size: 2000
- inline-max-per-routine: 10000
- inline-max-per-compile: 500000

Begin optimization report for: main()

Report from: Interprocedural optimizations [ipo]

INLINE REPORT: (main()) [1] ./simd\_example.c(19,12)

Report from: OpenMP optimizations [openmp]

OpenMP Construct at ./simd\_example.c(50,5)  
remark #16204: OpenMP multithreaded code generation for SINGLE was successful  
OpenMP Construct at ./simd\_example.c(49,5)  
remark #16201: OpenMP DEFINED REGION WAS PARALLELIZED  
OpenMP Construct at ./simd\_example.c(69,5)  
remark #16204: OpenMP multithreaded code generation for SINGLE was successful  
OpenMP Construct at ./simd\_example.c(68,5)  
remark #16201: OpenMP DEFINED REGION WAS PARALLELIZED

Report from: Loop nest, Vector & Auto-parallelization optimizations [loop, vec, par]

LOOP BEGIN at ./simd\_example.c(32,5)  
<Peeled loop for vectorization>  
LOOP END

LOOP BEGIN at ./simd\_example.c(32,5)  
remark #25084: Preprocess Loopnests: Moving Out Store [ ./simd\_example.c(32,18) ]  
remark #15300: LOOP WAS VECTORIZED  
LOOP END

LOOP BEGIN at ./simd\_example.c(32,5)  
<Alternate Alignment Vectorized Loop>  
LOOP END

LOOP BEGIN at ./simd\_example.c(32,5)  
<Remainder loop for vectorization>  
LOOP END

LOOP BEGIN at ./simd\_example.c(37,5)  
<Peeled loop for vectorization>  
LOOP END

LOOP BEGIN at ./simd\_example.c(37,5)

```

    remark #25084: Preprocess Loopnests: Moving Out Store    [
./simd_example.c(37,23) ]
    remark #15300: LOOP WAS VECTORIZED
LOOP END

LOOP BEGIN at ./simd_example.c(37,5)
<Alternate Alignment Vectorized Loop>
LOOP END

LOOP BEGIN at ./simd_example.c(37,5)
<Remainder loop for vectorization>
    remark #15301: REMAINDER LOOP WAS VECTORIZED
LOOP END

LOOP BEGIN at ./simd_example.c(37,5)
<Remainder loop for vectorization>
LOOP END

LOOP BEGIN at ./simd_example.c(44,5)
<Peeled loop for vectorization>
LOOP END

LOOP BEGIN at ./simd_example.c(44,5)
    remark #25084: Preprocess Loopnests: Moving Out Store    [
./simd_example.c(44,18) ]
    remark #15300: LOOP WAS VECTORIZED
LOOP END

LOOP BEGIN at ./simd_example.c(44,5)
<Alternate Alignment Vectorized Loop>
LOOP END

LOOP BEGIN at ./simd_example.c(44,5)
<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 BEGIN at ./simd_example.c(63,5)
<Distributed chunk1>
    remark #25084: Preprocess Loopnests: Moving Out Store    [
./simd_example.c(63,18) ]
    remark #25426: Loop Distributed (2 way)
    remark #15301: PARTIAL LOOP WAS VECTORIZED
LOOP END

LOOP BEGIN at ./simd_example.c(63,5)
<Remainder loop for vectorization, Distributed chunk1>
    remark #15335: remainder loop was not vectorized: vectorization possible but
seems inefficient. Use vector always directive or -vec-threshold0 to override
LOOP END

LOOP BEGIN at ./simd_example.c(63,5)
<Distributed chunk2>
    remark #15301: PARTIAL LOOP WAS VECTORIZED
LOOP END

LOOP BEGIN at ./simd_example.c(63,5)
<Remainder loop for vectorization, Distributed chunk2>
    remark #15335: remainder loop was not vectorized: vectorization possible but
seems inefficient. Use vector always directive or -vec-threshold0 to override
LOOP END

LOOP BEGIN at ./simd_example.c(49,5)

```

<Peeled loop for vectorization>

LOOP END

LOOP BEGIN at ./simd\_example.c(49,5)

remark #15300: LOOP WAS VECTORIZED

LOOP END

LOOP BEGIN at ./simd\_example.c(49,5)

<Alternate Alignment Vectorized Loop>

LOOP END

LOOP BEGIN at ./simd\_example.c(49,5)

<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 BEGIN at ./simd\_example.c(68,5)

<Peeled loop for vectorization>

LOOP END

LOOP BEGIN at ./simd\_example.c(68,5)

remark #15301: SIMD LOOP WAS VECTORIZED

LOOP END

LOOP BEGIN at ./simd\_example.c(68,5)

<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

Report from: Code generation optimizations [cg]

./simd\_example.c(19,12):remark #34051: REGISTER ALLOCATION : [main]

./simd\_example.c:19

Hardware registers

Reserved	:	2[ rsp rip]
Available	:	39[ rax rdx rcx rbx rbp rsi rdi r8-r15 mm0-mm7 zmm0-zmm15]
Callee-save	:	6[ rbx rbp r12-r15]
Assigned	:	20[ rax rdx rcx rbx rsi rdi r8-r15 zmm0-zmm5]

Routine temporaries

Total	:	556
Global	:	160
Local	:	396
Regenerable	:	137
Spilled	:	8

Routine stack

Variables	:	72 bytes*
Reads	:	24 [7.71e-01 ~ 0.8%]
Writes	:	25 [1.83e+00 ~ 1.8%]
Spills	:	104 bytes*
Reads	:	23 [6.33e-01 ~ 0.6%]
Writes	:	23 [1.04e+00 ~ 1.0%]

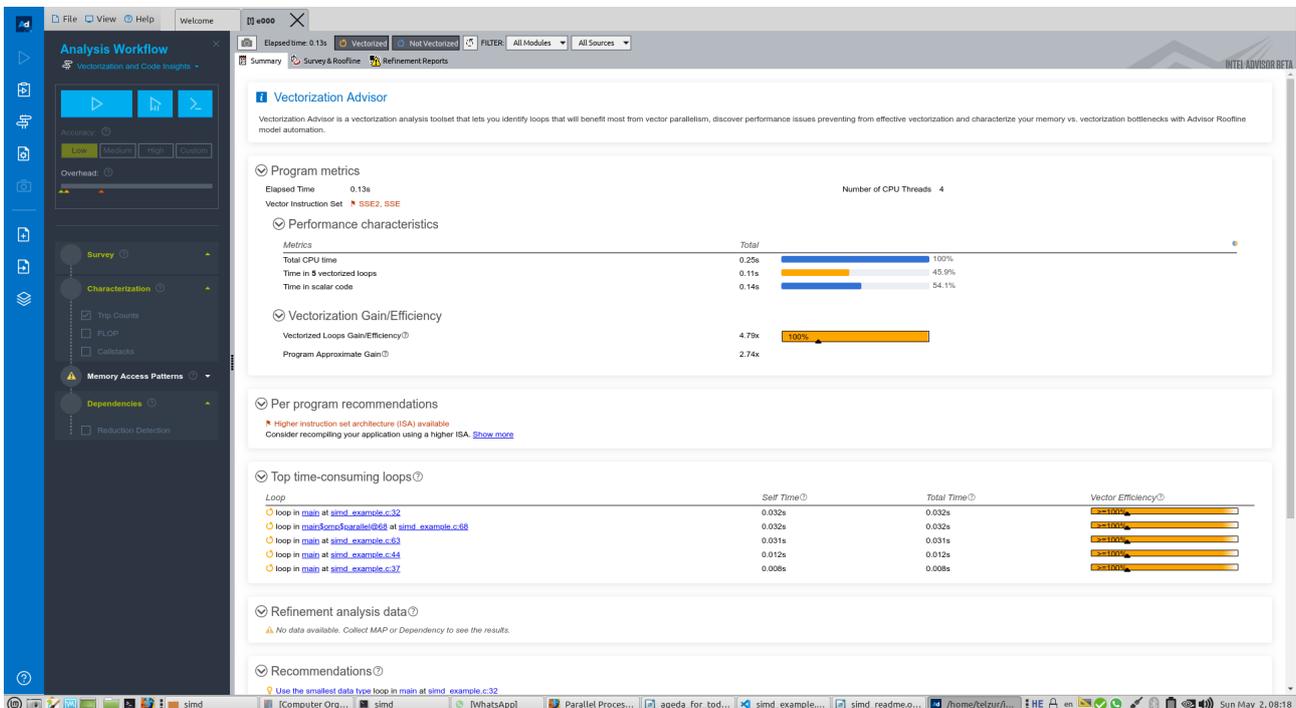
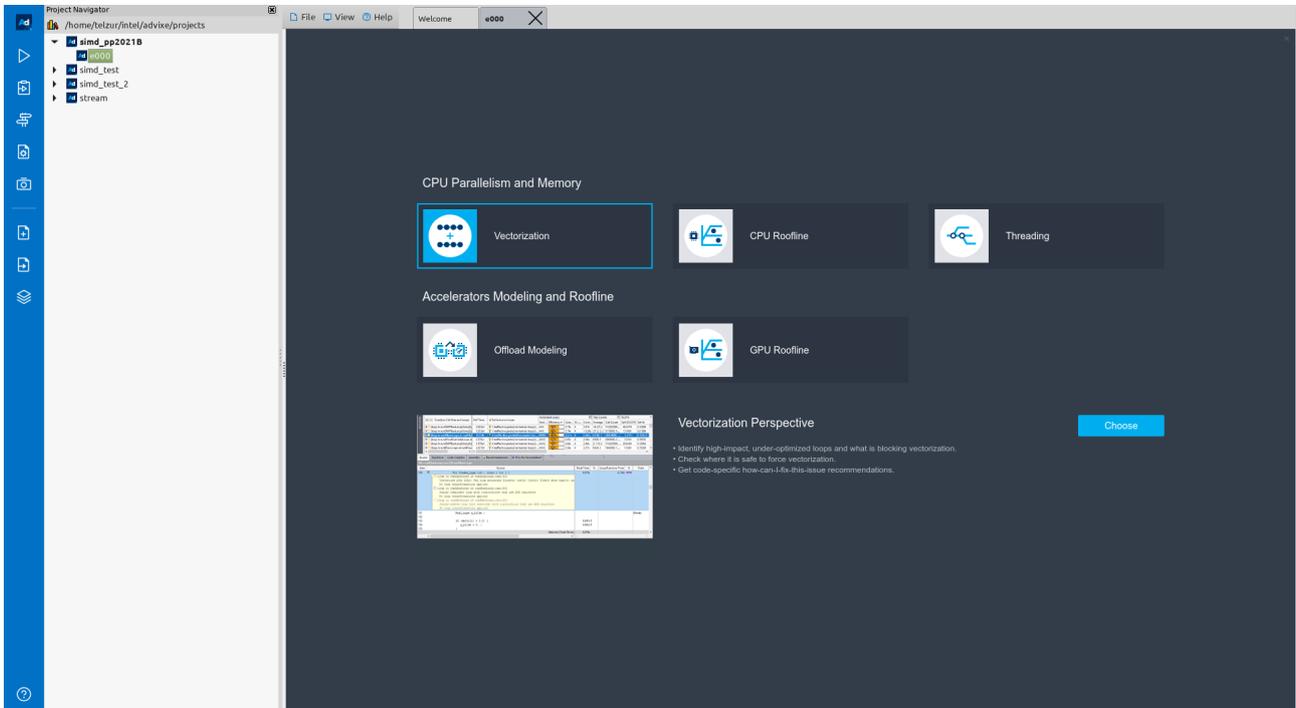
Notes

\*Non-overlapping variables and spills may share stack space, so the total stack size might be less than this.

## 8) Profile

### Intel Advisor:

~/science/opt/intel/oneapi/advisor/2021.1-beta10/bin64/advisor-gui



Intel Advisor Analysis Workflow: CPU Memory Roofline Insights

Summary: Higher instruction set architecture (ISA) available. Consider recompiling your application using a higher ISA.

Function Call Sites and Loops	Performance Issues	CPU Time	Type	Why No Vectorization?	Vectorized Loops	Instruction Set Analysis	Advanced	Location	
[Loop in main at simd_example.c:68]	1 Data type co.	0.046s	0.046s	Vectorized-Three	SSE2	4.94x	4	Type Conversions, Float32, Unrolled by 2	simd_example.c:68
[Loop in main at simd_example.c:32]	1 Data type co.	0.036s	0.036s	Vectorized-Body	SSE2	4.57x	4	Type Conversions, Float32, Unrolled by 2	simd_example.c:32
[Loop in main at simd_example.c:63]	1 Data type co.	0.027s	0.027s	Vectorized-Body	SSE, SSE2	5.04x	4	Type Conversions, Float32, Unrolled by 2	simd_example.c:63
[Loop in main at simd_example.c:49]	1 Data type co.	0.020s	0.020s	Vectorized-Three	SSE2	4.90x	4	Type Conversions, Float32, Unrolled by 2	simd_example.c:49
[Loop in main at simd_example.c:44]	1 Data type co.	0.018s	0.018s	Vectorized-Body	SSE2	4.18x	4	Type Conversions, Float32, Unrolled by 2	simd_example.c:44
[Loop in main at simd_example.c:37]	1 Data type co.	0.012s	0.012s	Vectorized-Body	SSE2	4.90x	4	Type Conversions, Float32, Unrolled by 2	simd_example.c:37
f_start	0.176s	0.000s	Function					NT-stores, Type Conversions, Float32, Unrolled by 2	simd_example.c:19
f_main	0.220s	0.000s	Function					Type Conversions, Unpacks, Float32, Unrolled by 2	simd_example.c:49
f_mainompparallel@68	0.046s	0.000s	Function					Type Conversions, Unpacks, Float32, Unrolled by 2	simd_example.c:68

Source Code Snippet:

```

53 printf("parallel loop\n");
54 printf("compute time%f\n",t_end-t_start);
55
56 //-----parallel for + simd-----
57 free(A);
58 free(B);
59 A = (float*) m_malloc(n * sizeof(float),64);
60 B = (float*) m_malloc(n * sizeof(float),64);
61
62 for (i=0;i<n;i++) { // initialize arrays
63     [Loop in main at simd_example.c:63]
64     Vectorized SSE loop processes Float32 data type(s)
65     Loop was distributed, chunk 1; loop was unrolled by 2
66     [Loop in main at simd_example.c:63]
67     Vectorized SSE2 loop processes Float32; Int32 data type(s) and includes Type Conversions
68     Loop was distributed, chunk 2; loop was unrolled by 2
69     [Loop in main at simd_example.c:63]
70     Scalar remainder loop [not executed]. Not vectorized; vectorization possible but seems inefficient. Use vector always directive or -vec-threshold to override
71     Loop was distributed, chunk 1
72     [Loop in main at simd_example.c:63]
73     Scalar remainder loop [not executed]. Not vectorized; vectorization possible but seems inefficient. Use vector always directive or -vec-threshold to override
74     Loop was distributed, chunk 2
75 }
76 t_start = omp_get_wtime();
77 #pragma omp parallel for simd
78 for(i = 0; i < n; i++)
79     B[i] = A[i]*A[i];
80 t_end = omp_get_wtime();

```

## Roofline report:

Intel Advisor Analysis Workflow: CPU Memory Roofline Insights

Summary: Higher instruction set architecture (ISA) available. Consider recompiling your application using a higher ISA.

Physical Cores: 4 | App Threads: 4 | Self Elapsed Time: 0.012 s | Total Elapsed Time: 0.012 s

Line	Source	Total Time	Loop/Function Time	%	Traits
53	printf("parallel loop\n");				
54	printf("compute time%f\n",t_end-t_start);				
55					
56	//-----parallel for + simd-----				
57	free(A);				
58	free(B);				
59	A = (float*) m_malloc(n * sizeof(float),64);				
60	B = (float*) m_malloc(n * sizeof(float),64);				
61					
62	for (i=0;i<n;i++) { // initialize arrays				
63	[Loop in main at simd_example.c:63]				
64	A[i]=i;				
65	B[i]=i;				
66	}				
67	t_start = omp_get_wtime();				
68	#pragma omp parallel for simd				