Code Optimization and the Roofline model demo

Document version: 2
Last update: 1.7.2024
Computer: TUF (core i9, Ubuntu 2024.04)
Intel OneAPI version on my computer:
$ icx -v
Intel(R) oneAPI DPC++/C++ Compiler 2024.1.0 (2024.1.0.20240308)

Reference:
https://tel-zur.net/blog/2024/03/20/the-roofline-model/

Folder:
/Teaching/PP/lectures/09/code/simd

Prepare the compilation environment

```
$ source /opt/intel/oneapi/setvars.sh

:: initializing oneAPI environment ...
   bash: BASH_VERSION = 5.2.21(1)-release
   args: Using "$@" for setvars.sh arguments:
:: advisor -- latest
:: ccl -- latest
:: compiler -- latest
:: dal -- latest
:: debugger -- latest
:: dev-utilities -- latest
:: dnnl -- latest
:: dpcpp-ct -- latest
:: dpl -- latest
:: ipp -- latest
:: ippcp -- latest
:: mkl -- latest
:: mpi -- latest
:: tbb -- latest
:: vtune -- latest
:: oneAPI environment initialized ::
```

Prepare the profiling environment:

```
cat /proc/sys/kernel/yama/ptrace_scope
// default value
1
$ echo "0"|sudo tee /proc/sys/kernel/yama/ptrace_scope   // change
it to 0
[sudo] password for telzur:
0
$ cat /proc/sys/kernel/yama/ptrace_scope
// verify
0
```

Compile:

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

Macro:

```
run_me.sh
```

Execute:

```
export OMP_NUM_THREADS=8

$ ./run_me.sh
Building a reference version

Building an optimized version

Running the reference version:
Serial computation
compute time=0.018967


Running the optimized version:
Parallel loop (OpenMP)
compute time=0.007473

Parallel simd
compute time=0.005096

Parallel loop+simd
compute time=0.002709
```

Check the optimization report

$ nano ./simd_intel.optrpt &

(partial display)

```
Global optimization report for : main

LOOP BEGIN at ./simd_intel.c (43, 5)
    remark #15335: loop was not vectorized: vectorization possible but seems inefficient. Use
vector always directive or -vec-threshold0 to override
    remark #25438: Loop unrolled without remainder by 8
LOOP END

LOOP BEGIN at ./simd_intel.c (62, 5)
<Multiversioned v2>
    remark #15319: Loop was not vectorized: novector directive used
LOOP END

LOOP BEGIN at ./simd_intel.c (62, 5)
<Multiversioned v1>
    remark #25228: Loop multiversioned for Data Dependence
    remark #25408: memset generated
    remark #15335: loop was not vectorized: vectorization possible but seems inefficient. Use
vector always directive or -vec-threshold0 to override
    remark #25438: Loop unrolled without remainder by 8
LOOP END

LOOP BEGIN at ./simd_intel.c (67, 5)
    remark #15301: SIMD LOOP WAS VECTORIZED
    remark #15305: vectorization support: vector length 4
LOOP END

LOOP BEGIN at ./simd_intel.c (84, 5)
<Multiversioned v2>
    remark #15319: Loop was not vectorized: novector directive used
LOOP END

LOOP BEGIN at ./simd_intel.c (84, 5)
<Multiversioned v1>
    remark #25228: Loop multiversioned for Data Dependence
    remark #25408: memset generated
    remark #15335: loop was not vectorized: vectorization possible but seems inefficient. Use
vector always directive or -vec-threshold0 to override
    remark #25438: Loop unrolled without remainder by 8
LOOP END
===========================================================

Global optimization report for : main.extracted

LOOP BEGIN at ./simd_intel.c (48, 5)
    remark #30000: OpenMP: Outlined parallel loop
```

```
    remark #15300: LOOP WAS VECTORIZED
    remark #15305: vectorization support: vector length 4
LOOP END

LOOP BEGIN at ./simd_intel.c (48, 5)
<Remainder loop for vectorization>
LOOP END
================================================================

Global optimization report for : main.extracted.12

LOOP BEGIN at ./simd_intel.c (89, 5)
    remark #30000: OpenMP: Outlined parallel loop
    remark #15301: SIMD LOOP WAS VECTORIZED
    remark #15305: vectorization support: vector length 4
LOOP END

LOOP BEGIN at ./simd_intel.c (89, 5)
<Remainder loop for vectorization>
LOOP END
================================================================
```

Feature	Loop Vectorization	OpenMP SIMD
Mechanism	Automatic compiler optimization	Programmer directive
Guarantee	Not guaranteed, depends on compiler and loop structure	Suggests vectorization, but compiler has final say
Scope	Broader concept, can involve loop unrolling and other techniques	Specifically targets SIMD instructions

The stream benchmark

URL: https://www.cs.virginia.edu/stream/ref.html

What is STREAM?

The STREAM benchmark is a simple synthetic benchmark program that measures sustainable memory bandwidth (in MB/s) and the corresponding computation rate for simple vector kernels.

Local folder:

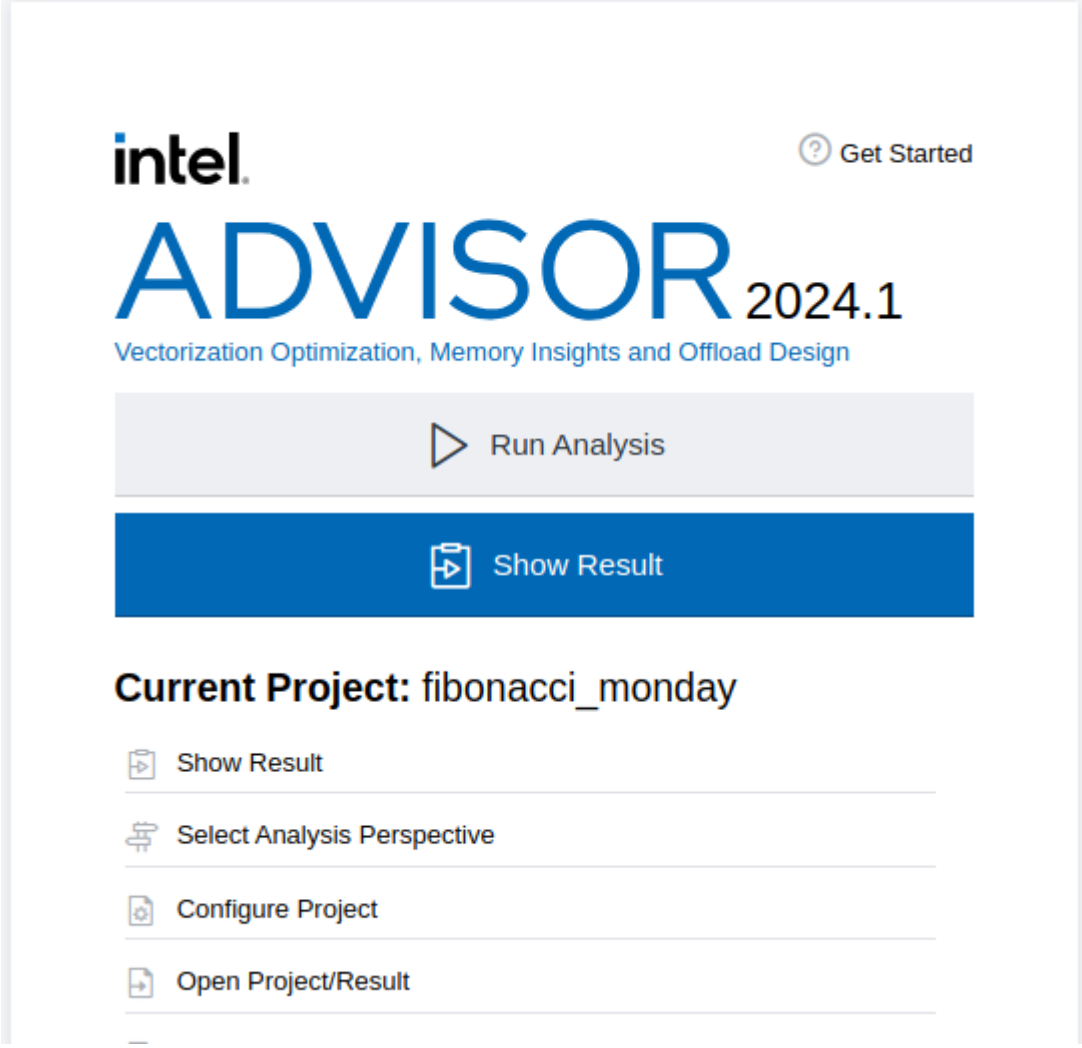
/Teaching/PP/lectures/09/code/stream

Macro:

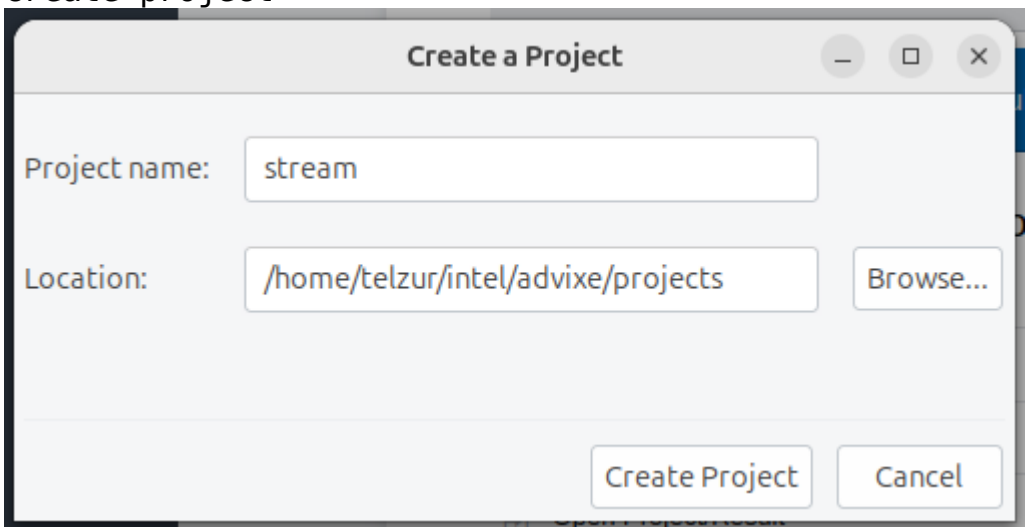
run_me.sh

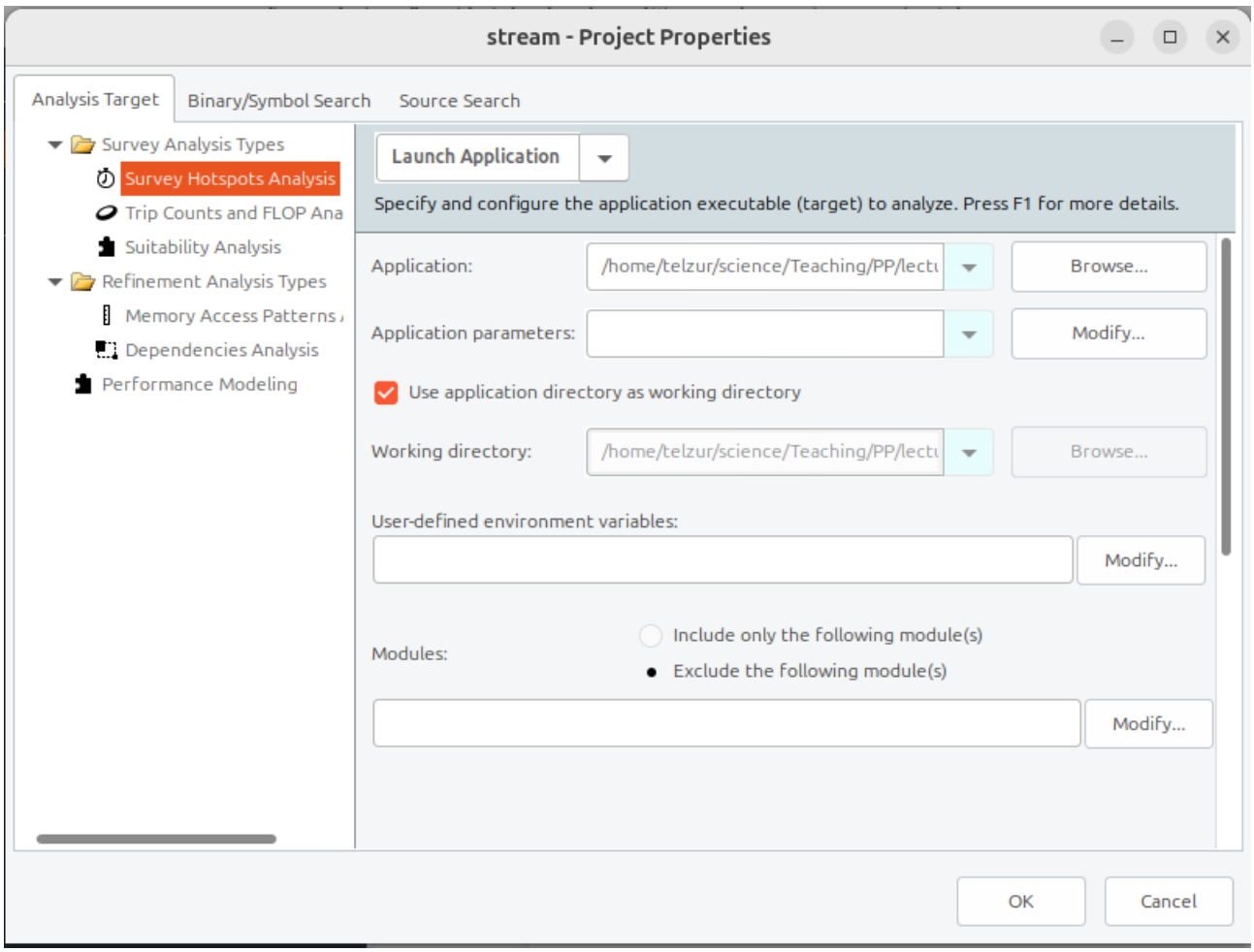
Profile

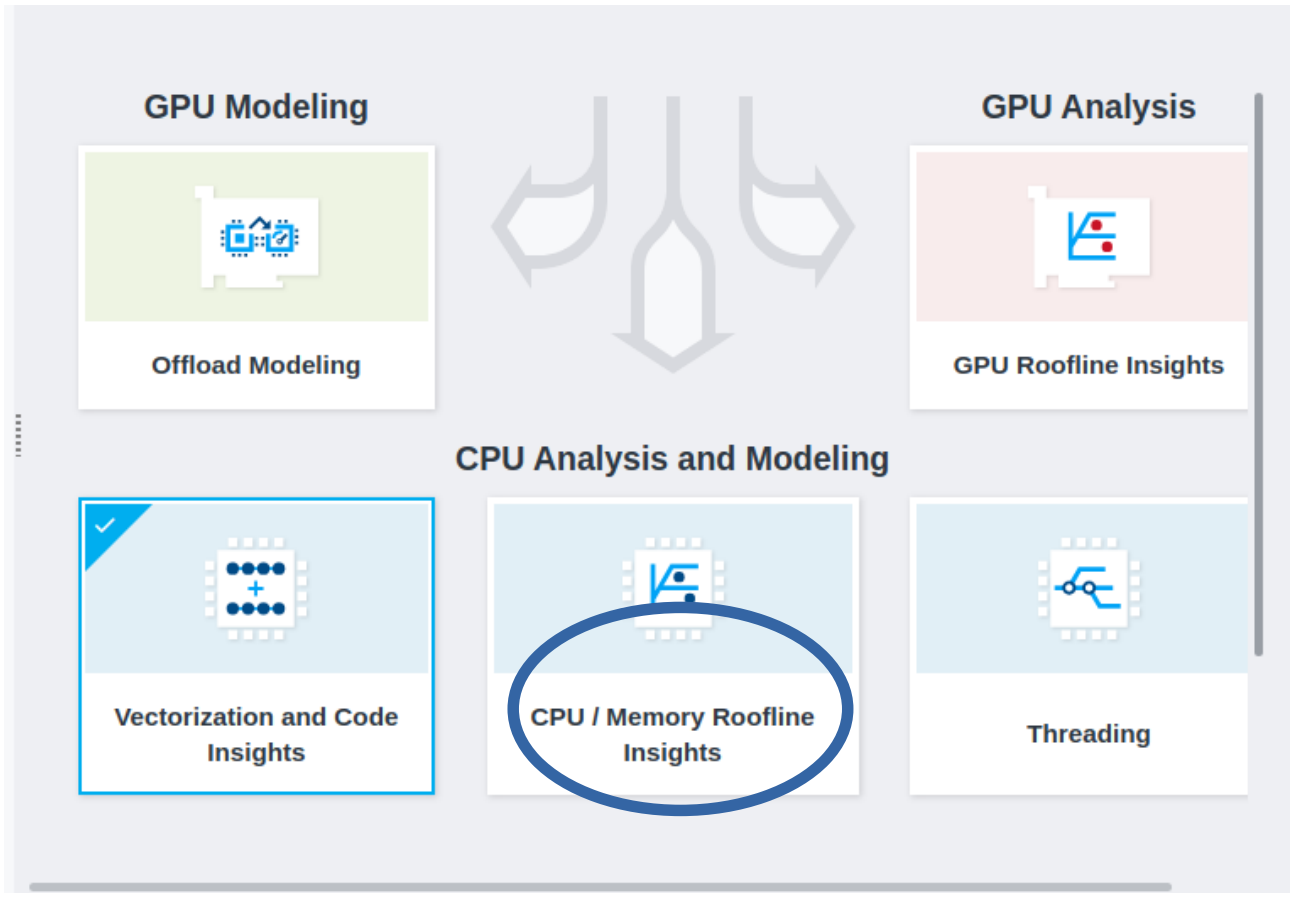
Intel Advisor: advisor-gui



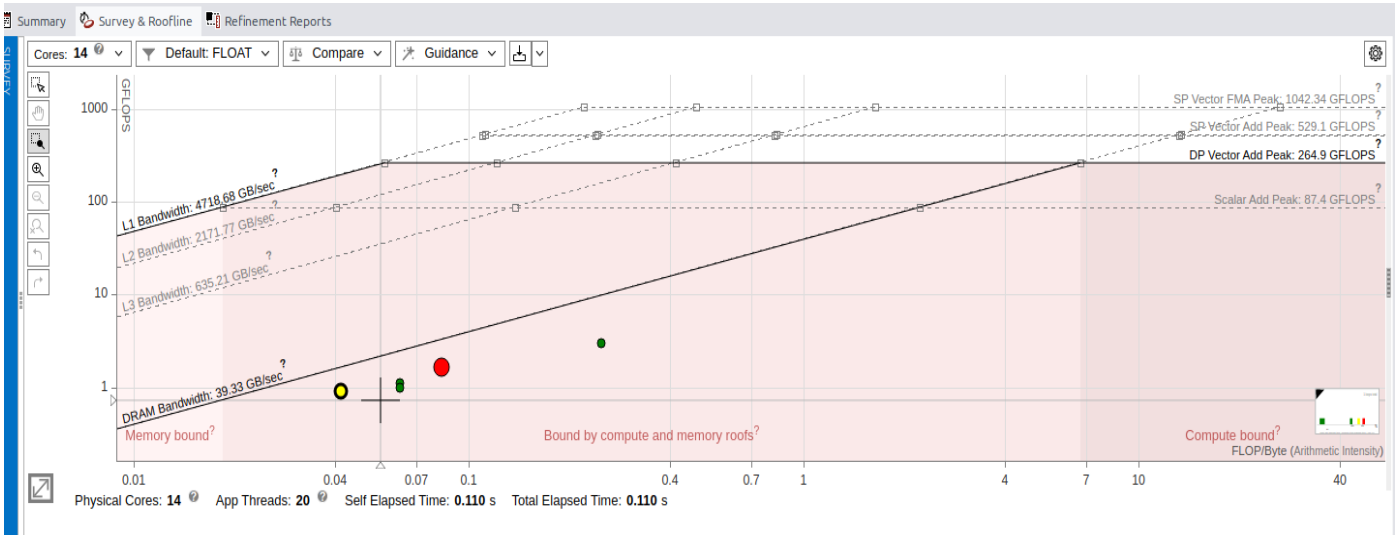
create_project



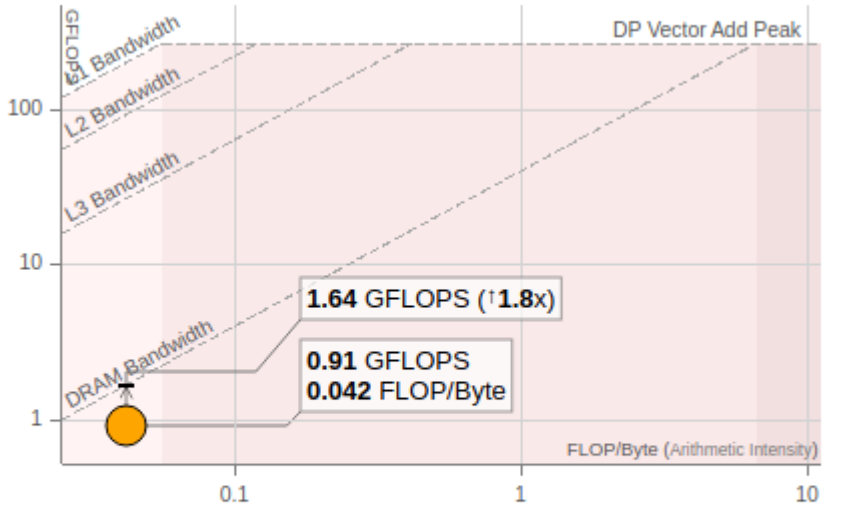

stream - Project Properties
Analysis Target
Binary/Symbol Search
Source Search
Survey Analysis Types
Survey Hotspots Analysis
Trip Counts and FLOP Ana
Suitability Analysis
Refinement Analysis Types
Memory Access Patterns
Dependencies Analysis
Performance Modeling
Launch Application
Specify and configure the application executable (target) to analyze. Press F1 for more details.
Application:
/home/telzur/science/Teaching/PP/lectu
Browse...
Application parameters:
Modify...
Use application directory as working directory
Working directory:
/home/telzur/science/Teaching/PP/lectu
Browse...
User-defined environment variables:
Modify...
Modules:
Include only the following module(s)
Exclude the following module(s)
Modify...
OK
Cancel


GPU Modeling
Offload Modeling
GPU Analysis
GPU Roofline Insights
CPU Analysis and Modeling
Vectorization and Code Insights
CPU / Memory Roofline Insights
Threading

```
334         for (j=0; j<STREAM_ARRAY_SIZE; j++)
335             c[j] = a[j]+b[j];
                [loop in main.extracted.78 at stream_intel.c:335]
                    Vectorized SSE2 loop processes Int64 data type(s)
336     #endif
337         times[2][k] = mysecond() - times[2][k];
338
339         times[3][k] = mysecond();
340     #ifdef TUNED
341             tuned_STREAM_Triad(scalar);
342     #else
343     #pragma omp parallel for
344         for (j=0; j<STREAM_ARRAY_SIZE; j++)
345             a[j] = b[j]+scalar*c[j];
                [loop in main.extracted.83 at stream_intel.c:345]
                    Vectorized SSE2 loop processes Int64 data type(s)
346     #endif
347         times[3][k] = mysecond() - times[3][k];
348         }
349
```



Roofline



This loop is mostly memory bound

The performance of the loop is bounded by the DRAM bandwidth.

You can switch to the Recommendations tab to see optimization recommendations in the **Roofline Conclusions** section.

==========That's it!==========