PlaFRIM

Courtès L., Rué F.

November 8, 2019

Table of contents

The hard way

- printf("%i",time(NULL));

The hard way

- printf("%i",time(NULL));
- 

The optimization objectives

- Improve the speed of execution

The optimization objectives

- Improve the speed of execution
- Reduce memory footprint

The optimization objectives

- Improve the speed of execution
- Reduce memory footprint
- Reduce energy consumption

The optimization objectives

- Improve the speed of execution
- Reduce memory footprint
- Reduce energy consumption
- Consume fewer resources

The Process

- Identify bottlenecks (Profiling)
- Choose better algorithms or improve implementation (Optimization)

How profilers do it

- Call stack sampling

How profilers do it

- Call stack sampling
- Optional function call instrumentation

How profilers do it

- Call stack sampling
- Optional function call instrumentation
- Hardware simulation

How profilers do it

- Call stack sampling
- Optional function call instrumentation
- Hardware simulation
- Hardware counter

Memory

Understanding memory locality

Storage Area	Register	L1 Cache	L2 Cache	RAM	Swap
Cycles to Access	≤ 1	≈ 3	≈ 14	≈ 240	$\approx 10^7$
Town	Talence	Pessac	Cestas	Toulouse	Mars

General Exploration

Optimization and granularity

The easiest way

- Time command
- Real, user & sys time
- Best way to evaluate scalability

The easiest way

- Time command
- Real, user & sys time
- Best way to evaluate scalability
- Accuracy of the evaluation?

Profiler

static instrumentation - gprof

- Sampling technique
- no instrumentation needed
- 2 types of view (flat profile and call graph)

Profiler

static instrumentation - gprof

- Sampling technique
- no instrumentation needed
- 2 types of view (flat profile and call graph)
- Annotated code

Profiler

static instrumentation - gprof

- use the -pg option to compile
- evaluate the output : gprof 'binary name' gmon.out

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 94.75      4.77     4.77       50    95.32    95.32  compute_wave
  3.98      4.97     0.20                             main
  1.39      5.04     0.07        2    35.04    35.04  vtkprint
  0.00      5.04     0.00       50     0.00     0.00  progress_bar

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.
```

Profiler

static instrumentation - gprof

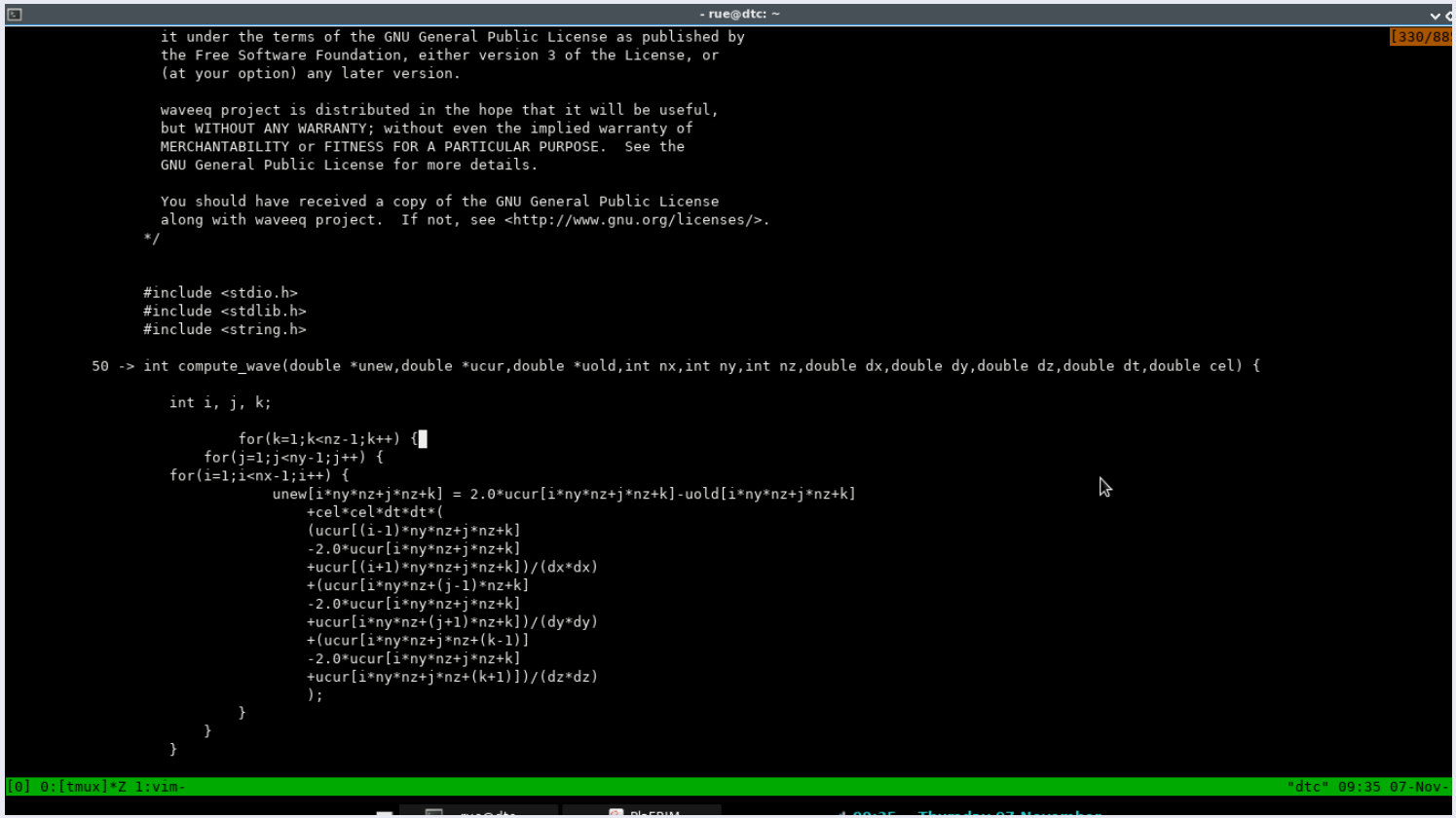
- use the -pg option to compile
- evaluate the output : gprof 'binary name' gmon.out

```
                     Call graph (explanation follows)

granularity: each sample hit covers 2 byte(s) for 0.20% of 5.04 seconds

index % time    self  children    called     name
                                                 <spontaneous>
[1]    100.0    0.20    4.84                 main [1]
                4.77    0.00      50/50          compute_wave [2]
                0.07    0.00       2/2           vtkprint [3]
                0.00    0.00      50/50          progress_bar [4]
-----------------------------------------------
                4.77    0.00      50/50          main [1]
[2]     94.6    4.77    0.00      50          compute_wave [2]
-----------------------------------------------
                0.07    0.00       2/2           main [1]
[3]      1.4    0.07    0.00       2          vtkprint [3]
-----------------------------------------------
                0.00    0.00      50/50          main [1]
[4]      0.0    0.00    0.00      50          progress_bar [4]
-----------------------------------------------

 This table describes the call tree of the program, and was sorted by
 the total amount of time spent in each function and its children.

 Each entry in this table consists of several lines.  The line with the
 index number at the left hand margin lists the current function.
 The lines above it list the functions that called this function,
 and the lines below it list the functions this one called.
 This line lists:
     index      A unique number given to each element of the table.
                Index numbers are sorted numerically.
```

Profiler

static instrumentation - gprof

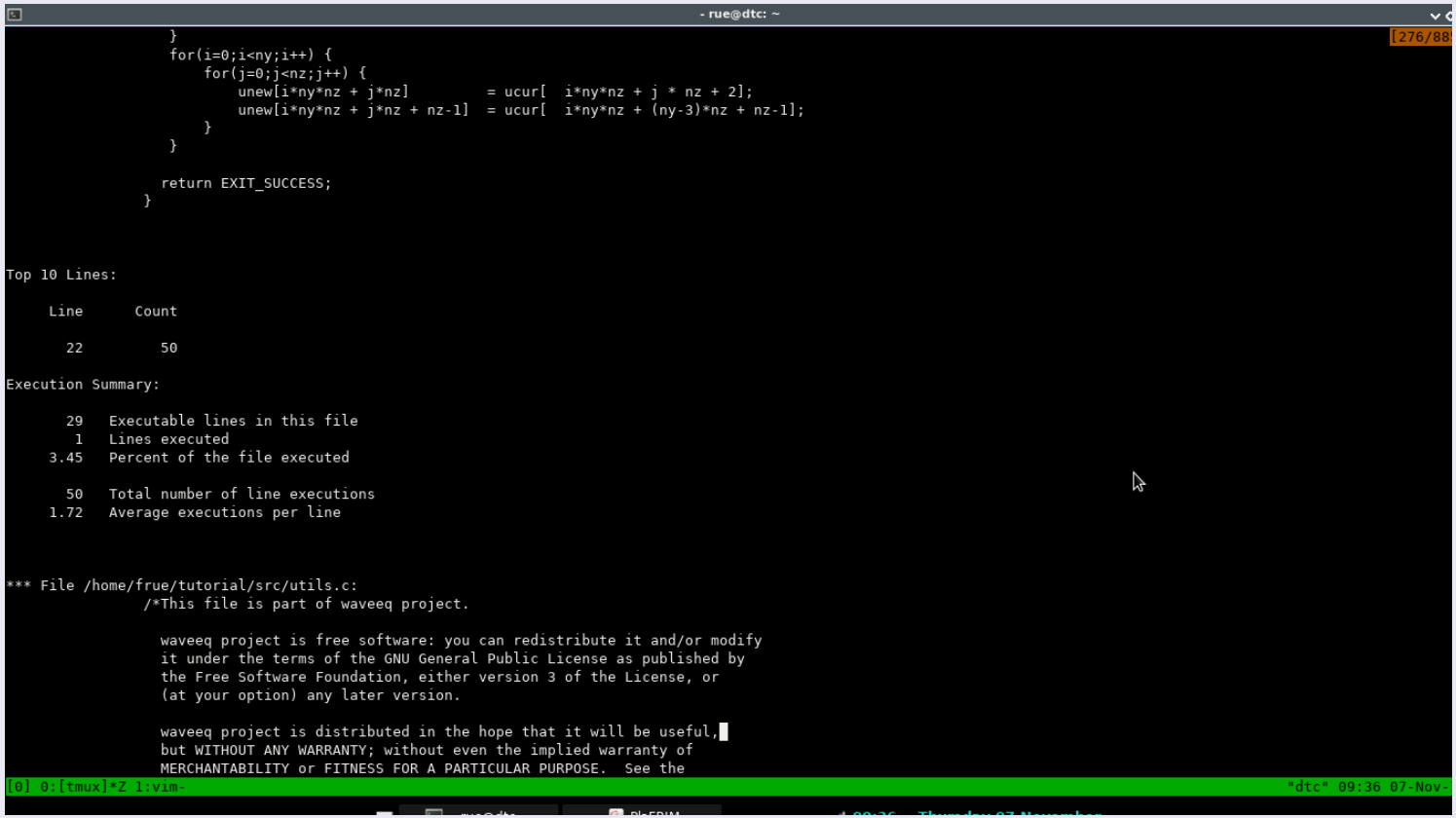
- gprof -A -l 'binary name' gmon.out



Profiler

static instrumentation - gprof

- gprof -A -l 'binary name' gmon.out



Profiler

- and for memory usage ?

Profiler

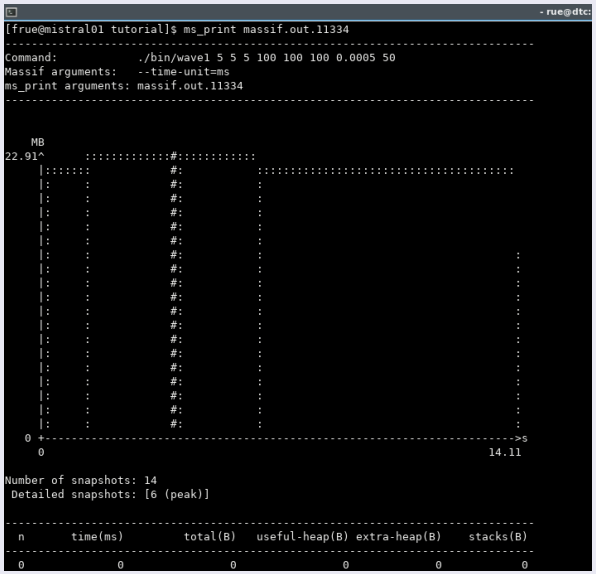
Dynamic instrumentation - valgrind

- Done at execution time
- no instrumentation needed
- different tools for differents analysis
  - massif - heap profiler
  - callgrind - call history among functions
  - cachegrind - interactions with machine cache

Profiler

Dynamic instrumentation - valgrind

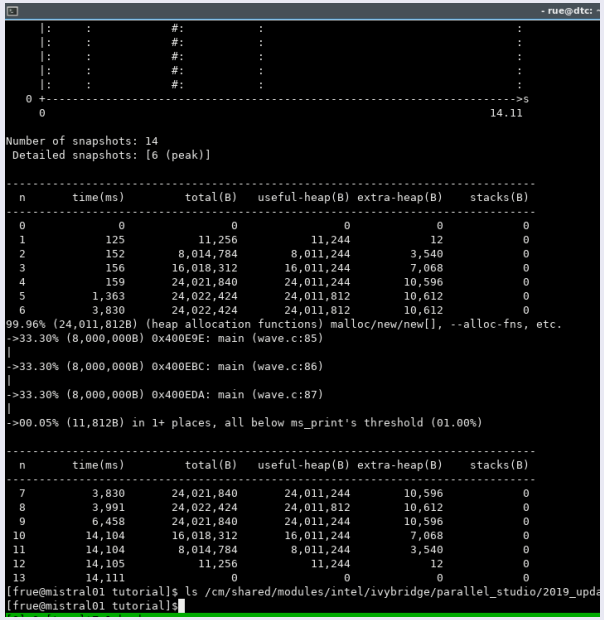
- valgrind –tool=massif –time-unit=ms ./bin/wave0 5 5 5 100 100 100 0.0005 50
- ms_print massif.out.%pid



Profiler

Dynamic instrumentation - valgrind

- valgrind –tool=massif –time-unit=ms ./bin/wave0 5 5 5 100 100 100 0.0005 50
- ms_print massif.out.%pid

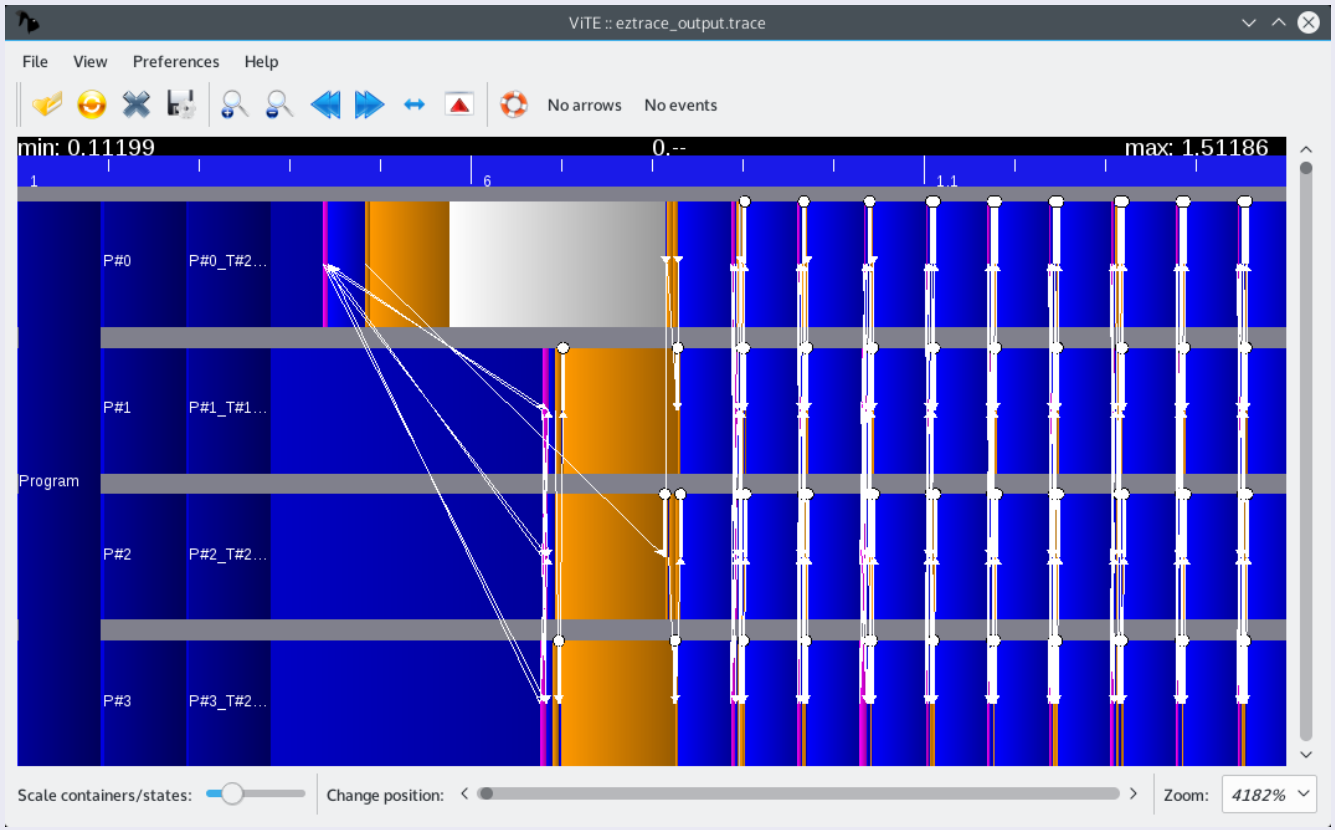


Profiler

- what kind of expertise ?

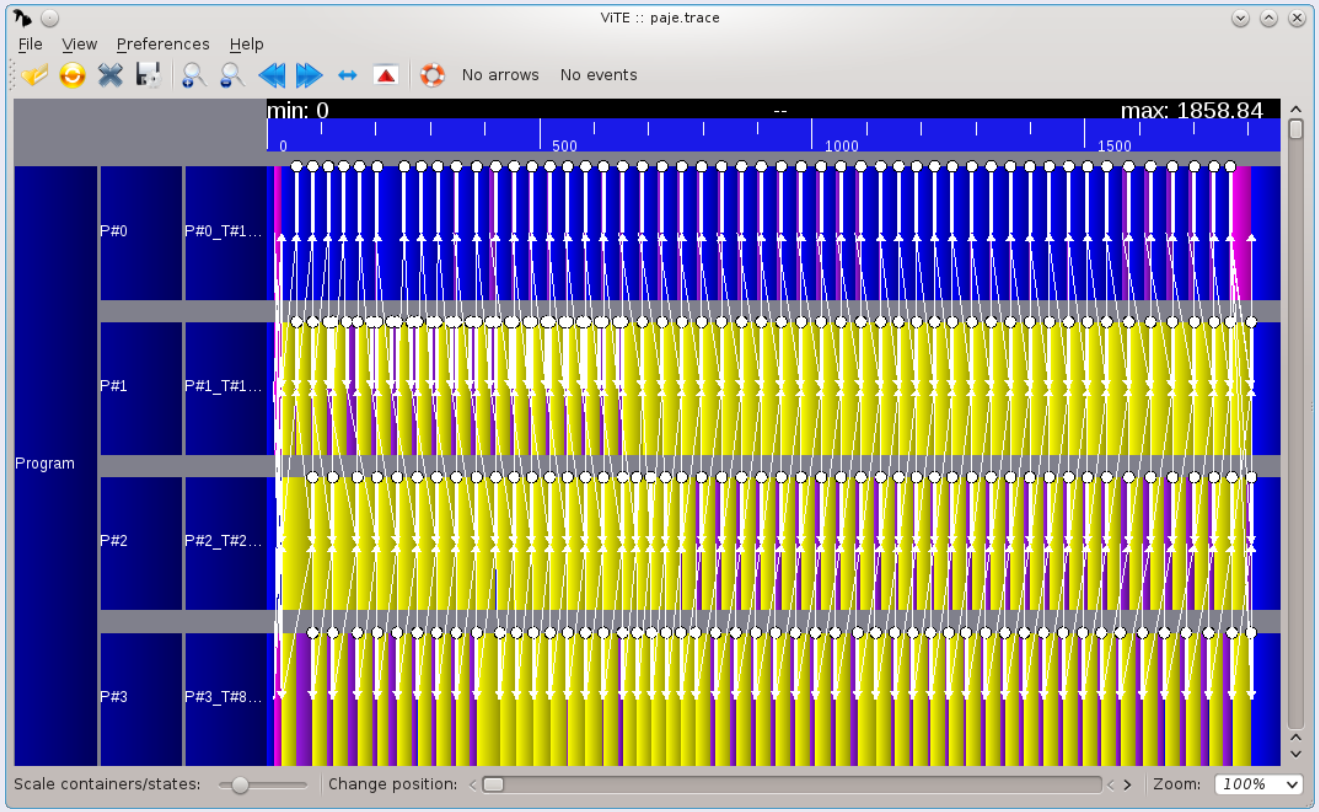
Profiler

- what kind of image of your program do you need ?



Profiler

- what kind of image of your program do you need ?



The Roofline model

roofline

The model

cache aware roofline model

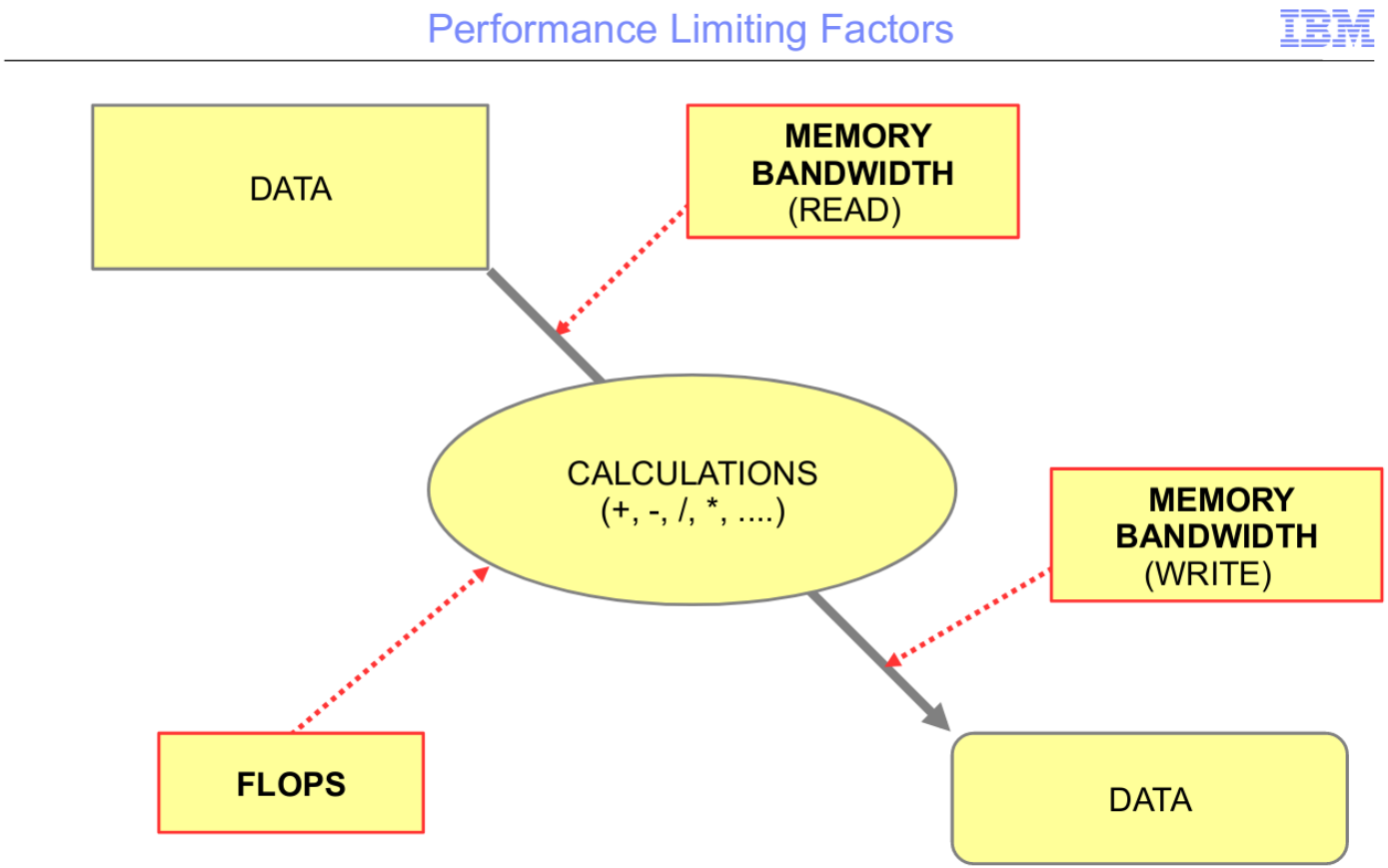


Figure: IBM - ICSC 2014, Shanghai, China

The model

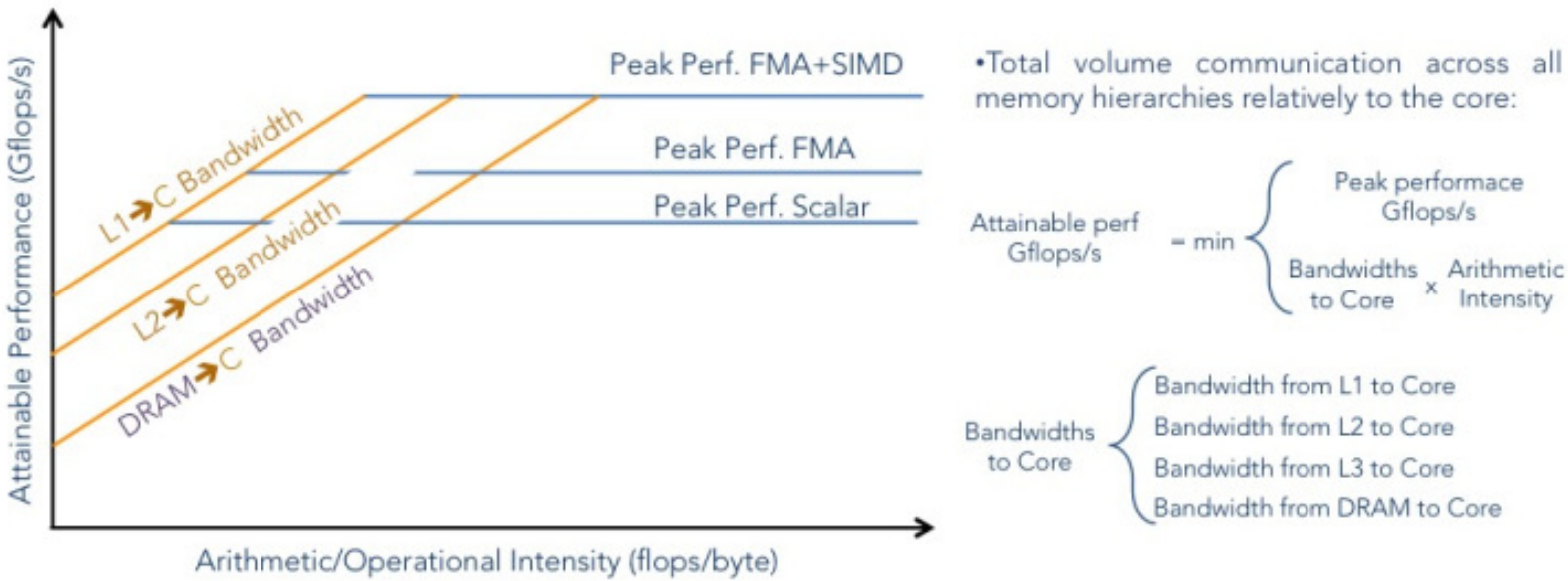


Figure: PICSAR Project

The model

- AXPY: y[i] = a*x[i] + y[i], a is real, i=0...N-1
- 2 Flops for each element of x & y.
  - well balanced: 1 multiply, 1 add
  - need to load x[i] and y[i] for each 'i': 2 x 4 = 8 bytes
    - keep 'a' in a register
  - need to write out y[i]: another 4 bytes
  - Arithmetic Intensity: 2 FLOPS/12 bytes = 1/6
  - Speed of light for performance (working from memory)
    - on an Intel Core i7 3960X with mem b/w of 51.2 GB/sec: 8.53 Gflops
      - even tho the socket has a peak speed of ***316.8 Gflops***
    - if x & y fit into caches, ***higher cache B/W*** results in ***higher performance***
    - on an NVIDIA M2090 GPU with mem b/w of 177 GB/sec: 29.5 Gflops
      - even tho GPU can do ***1.3 Teraflops***

Figure: Thomas Jefferson National Accelerator Facility

The model

cache aware roofline model

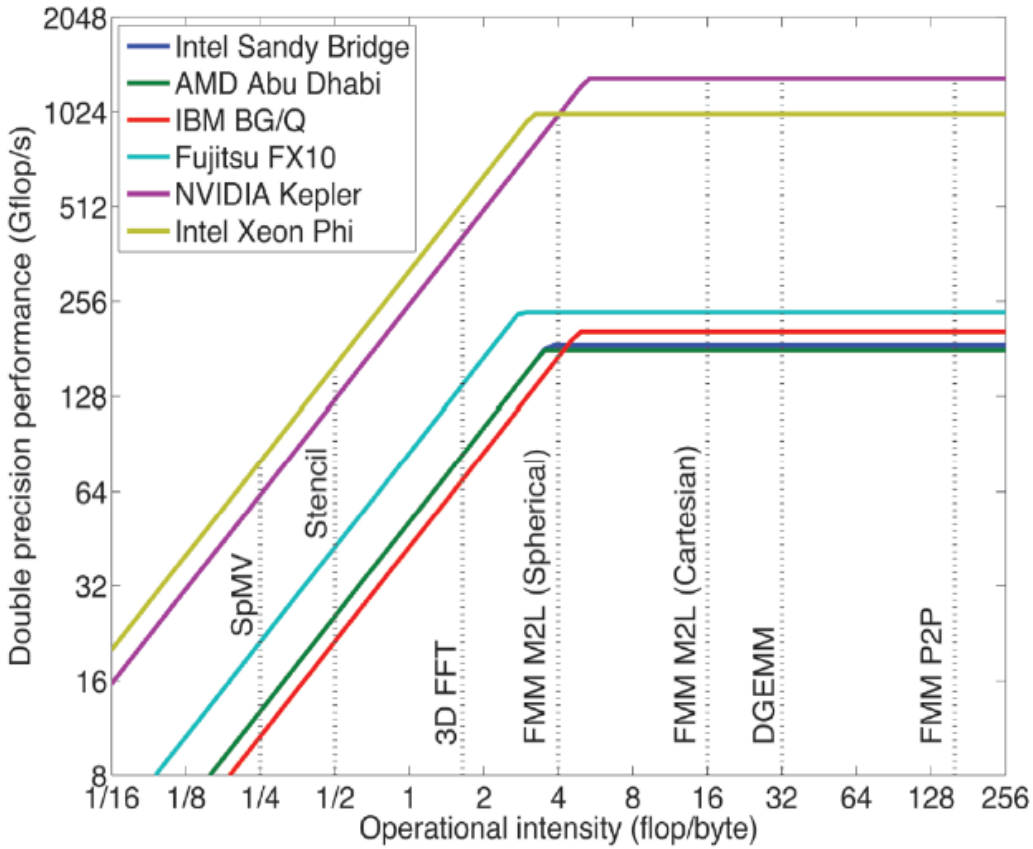


Figure: IBM - ICSC 2014, Shanghai, China

The model

- How to construct this model ?

The model

- How to construct this model ?
- How to evaluate your Arithmetic Intensity ?

Roofline evaluation

...

Roofline evaluation

evaluate the performance you can achieve

Performance achievement

Understanding memory locality

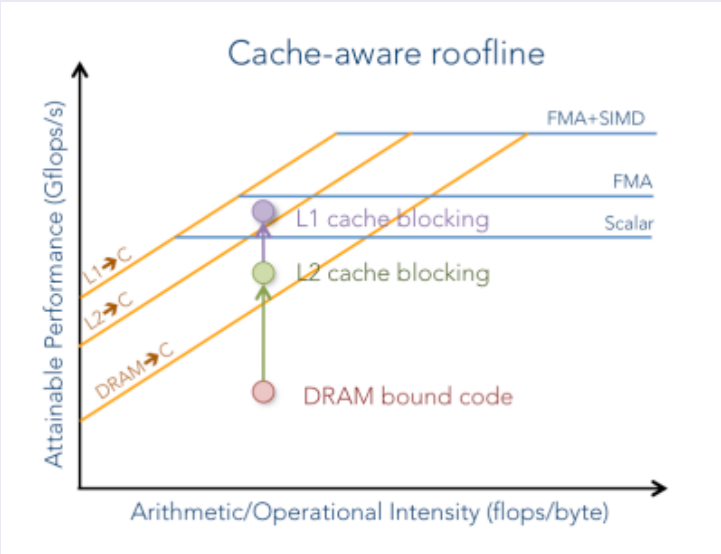


Figure: Memory Bound

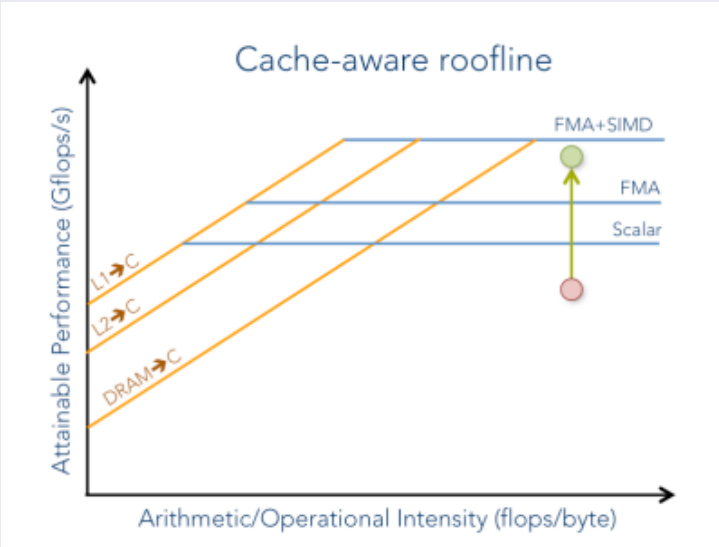


Figure: Compute Bound

Intel Advisor tool

one tool to do that ...

Intel Advisor tool

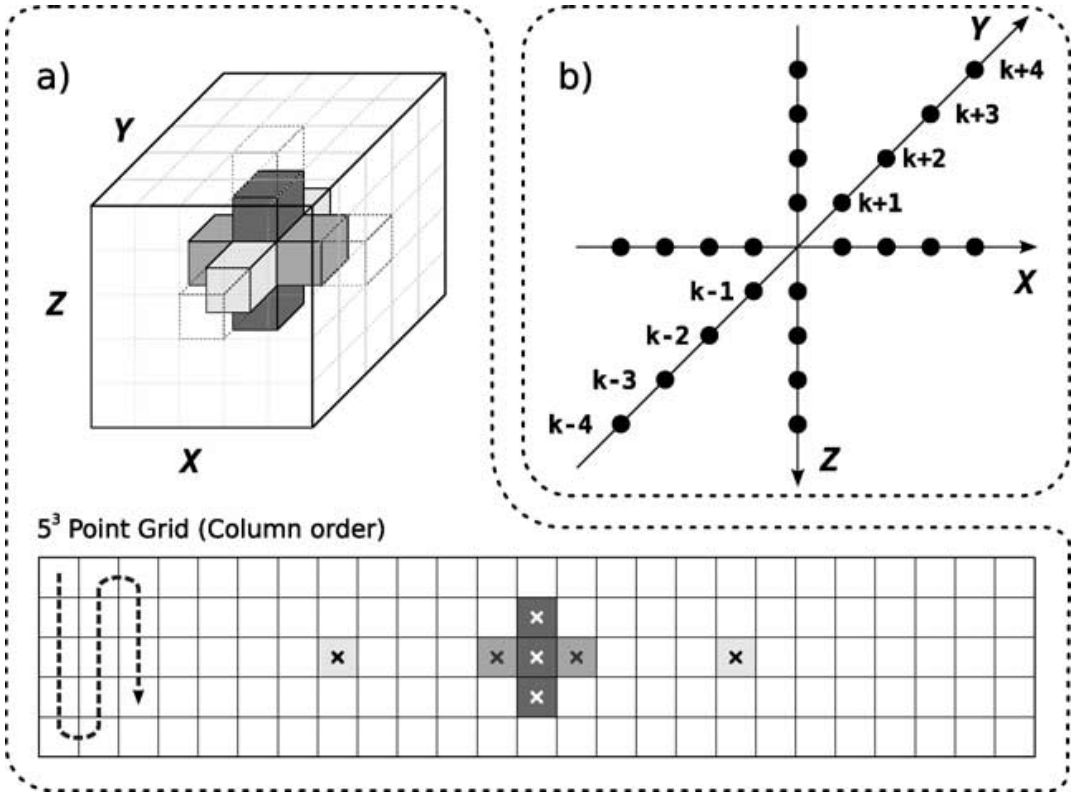


Figure: The 3D stencil: its memory access pattern (a) and the data points it uses (b). - Raul de la Cruz, BSC

Intel Advisor tool

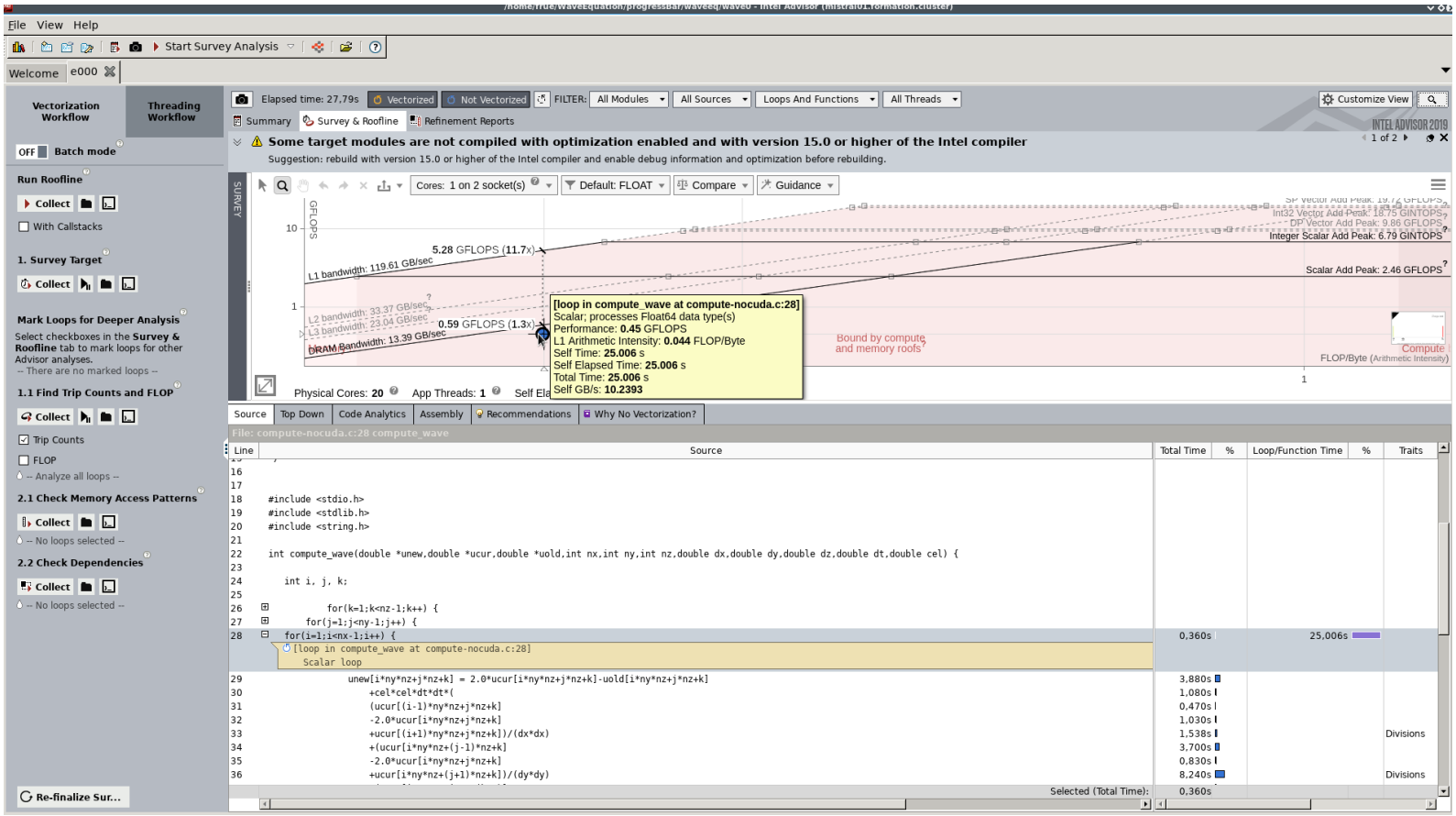


Figure: Stencil 1 thread - roofline

Intel Advisor tool

- module load compiler/gcc/9.1.0 compiler/intel/2019_update4 intel/vtune-advisor
- advixe-cl -collect roofline –project-dir=wave0 –ignore-checksums ./bin/wave0 5 5 5 100 100 100 0.0005 500
- advixe-gui

Intel Advisor tool

- module load compiler/gcc/9.1.0 compiler/intel/2019_update4 intel/vtune-advisor
- advixe-cl -collect roofline –project-dir=wave0 –ignore-checksums ./bin/wave0 5 5 5 100 100 100 0.0005 500
- advixe-gui
- RTFM : the README file

Intel Advisor tool

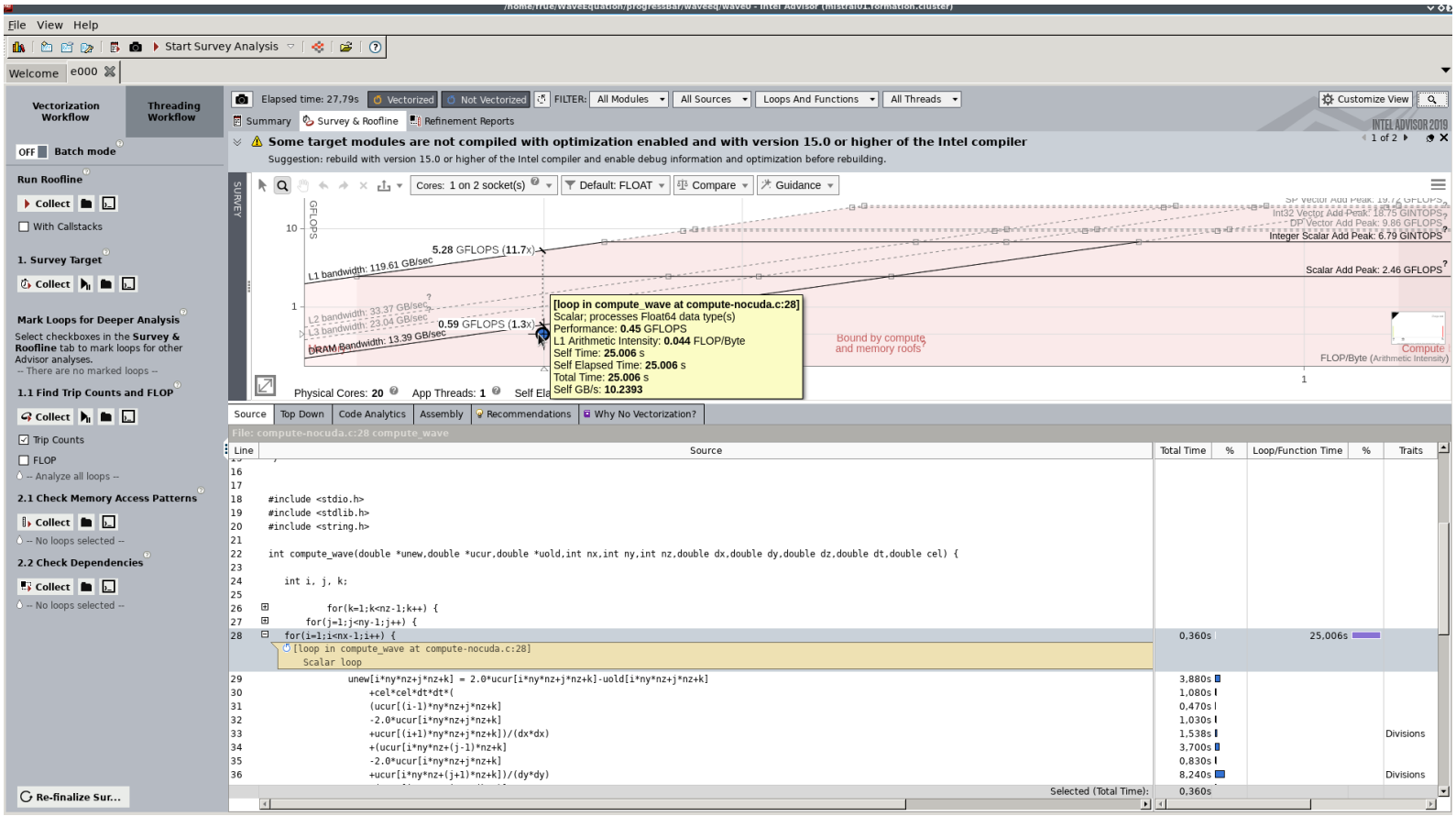


Figure: Stencil 1 thread - roofline

Intel Advisor tool

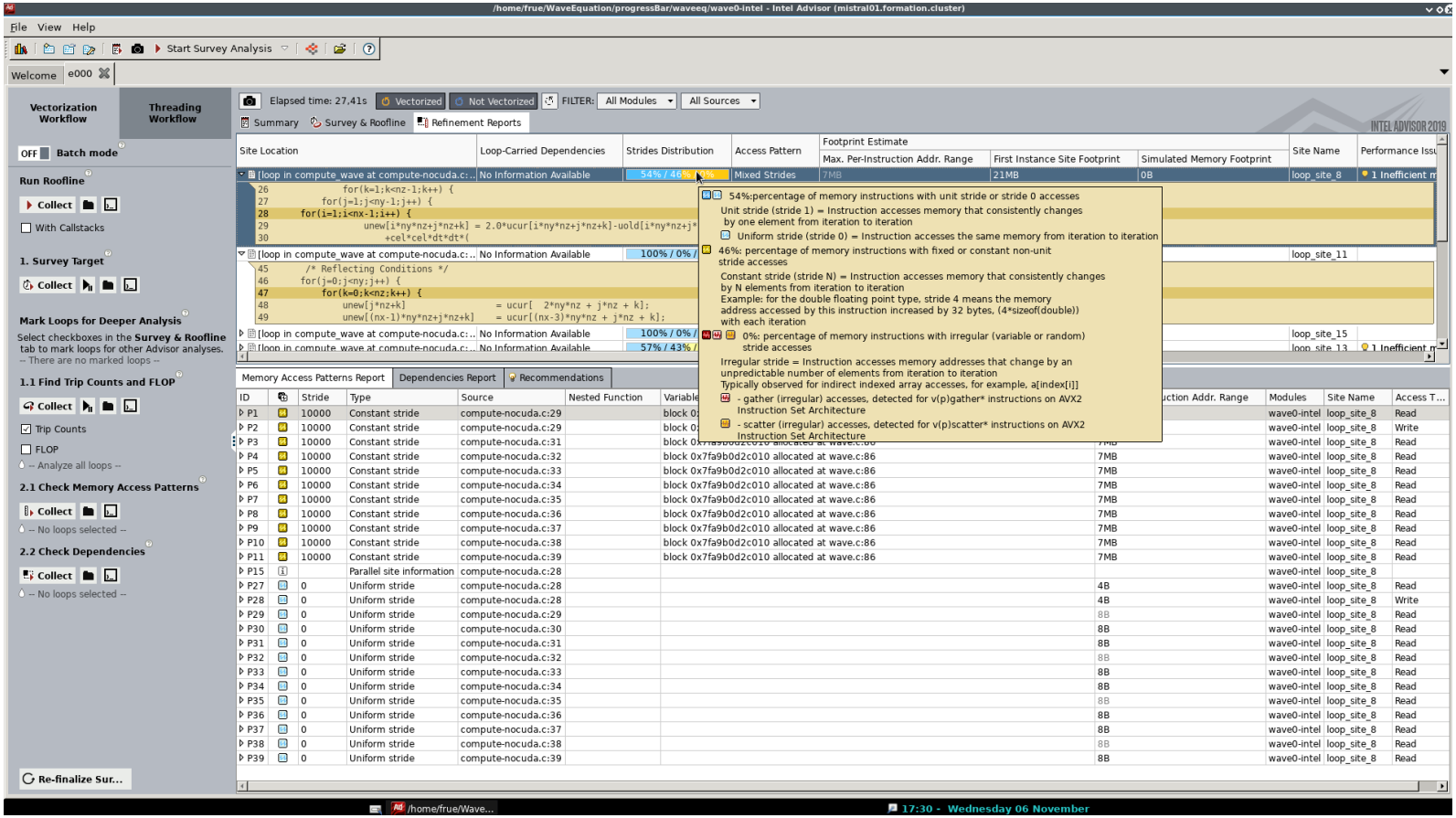


Figure: Stencil 1 thread - memory access pattern

Intel Advisor tool

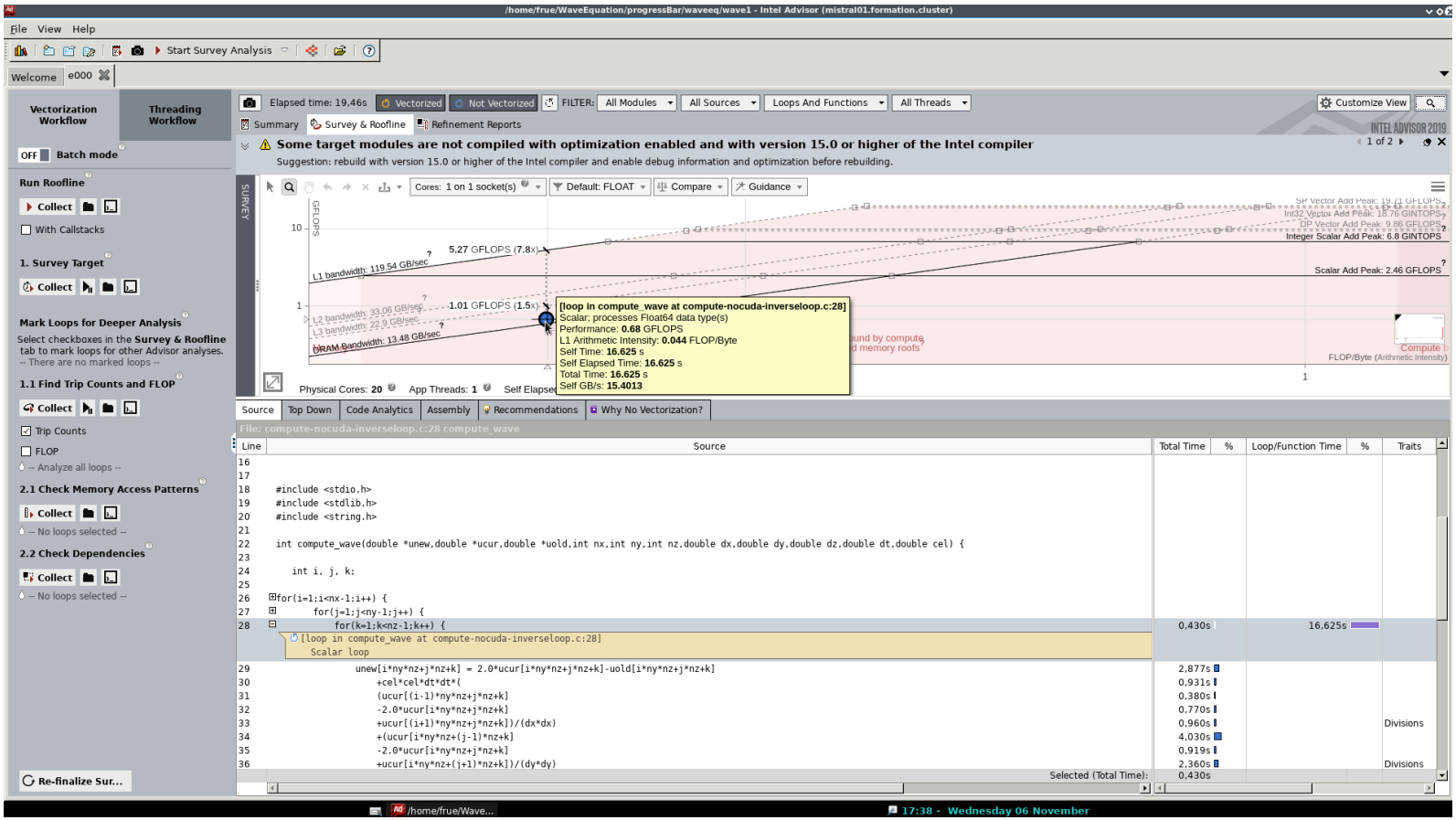


Figure: Stencil 1 thread - inverse loop - roofline

Intel Advisor tool

- strides distribution - better performance

Intel Advisor tool

- strides distribution - better performance
- cache blocking technic?

Intel Advisor tool

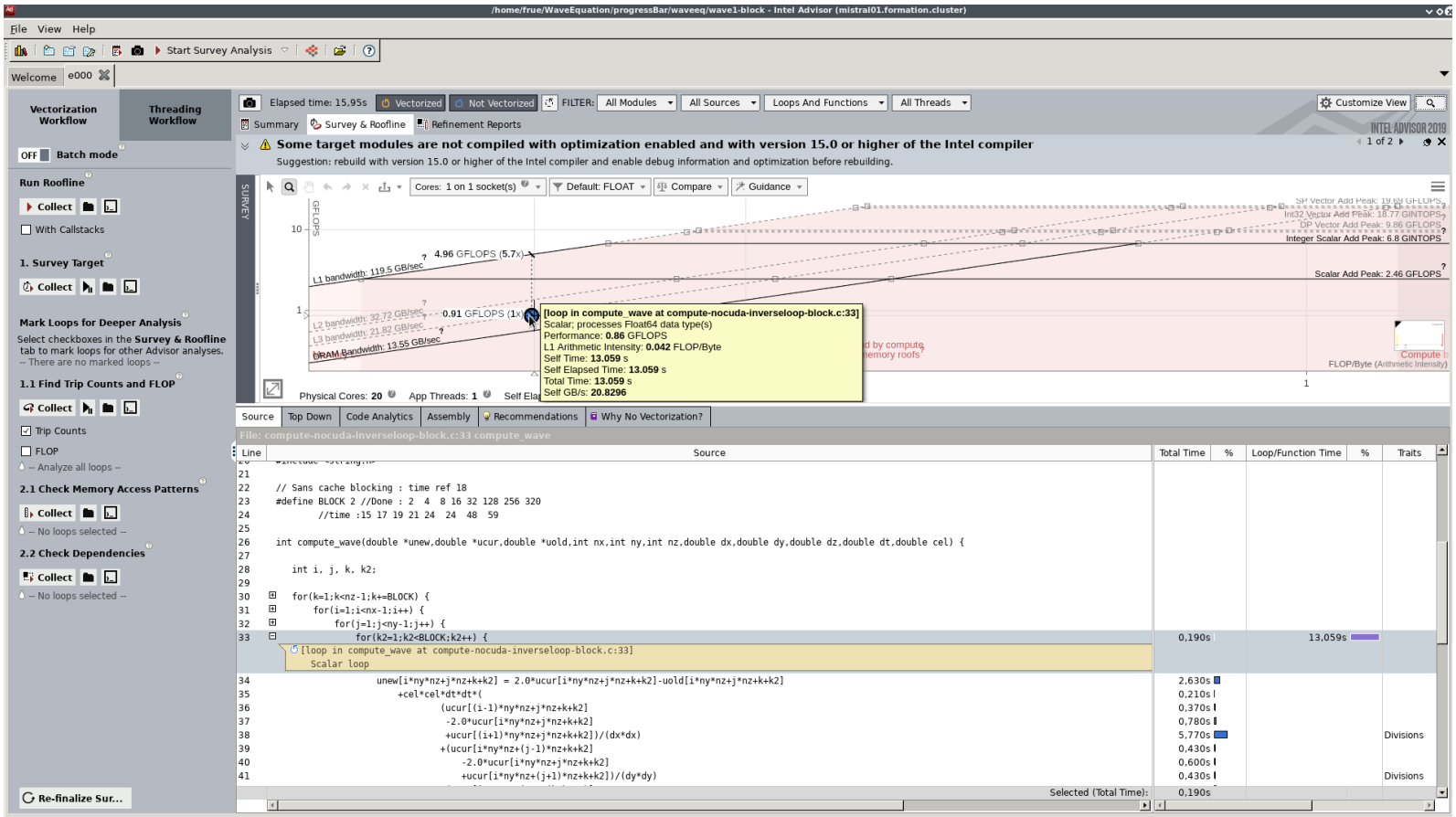


Figure: Stencil 1 thread - inverse loop & cache blocking - roofline

Intel Advisor tool

- OpenMP ?

Intel Advisor tool

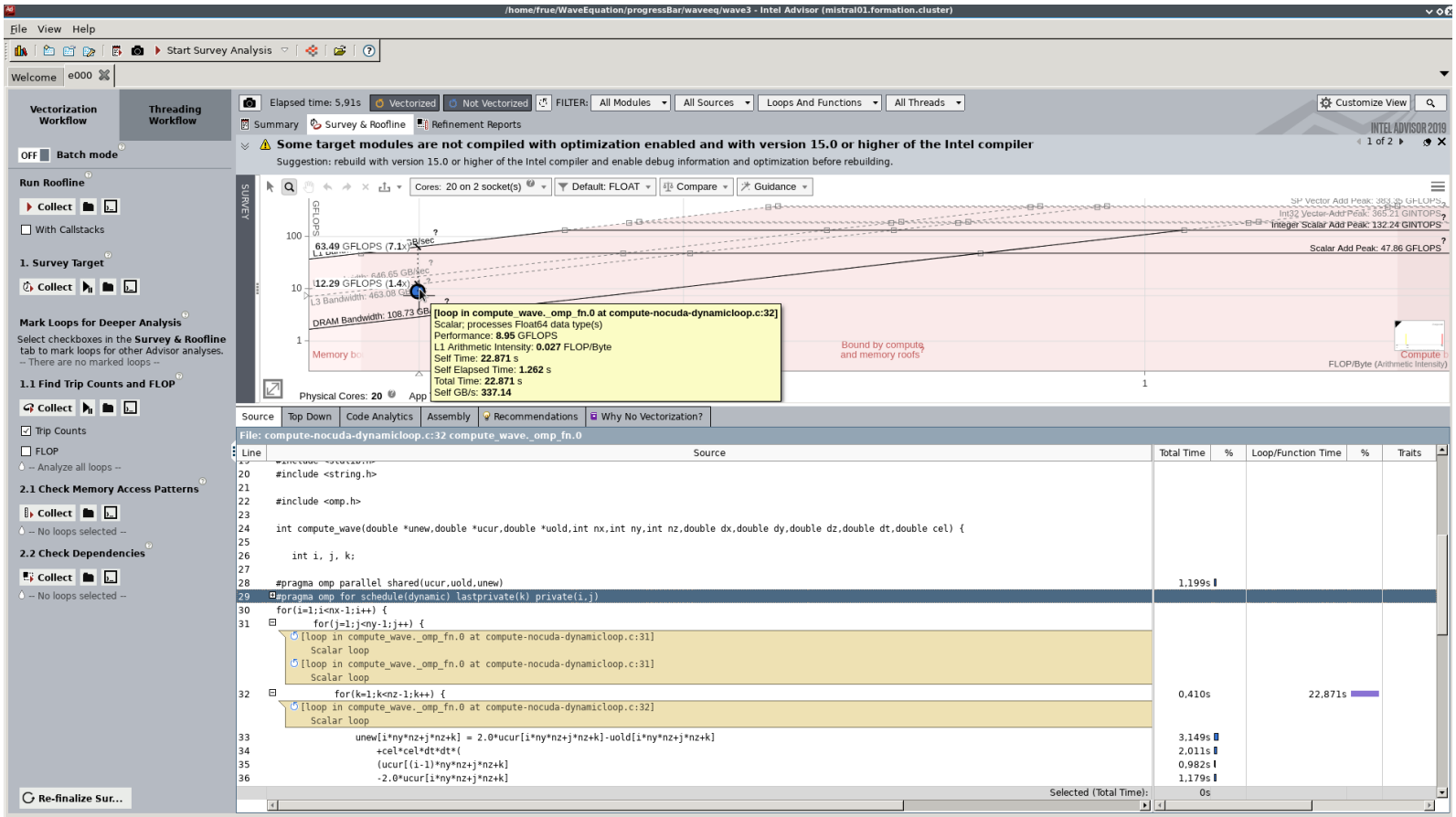


Figure: Stencil 20 threads - inverse loop & OpenMP - roofline

Intel Advisor tool

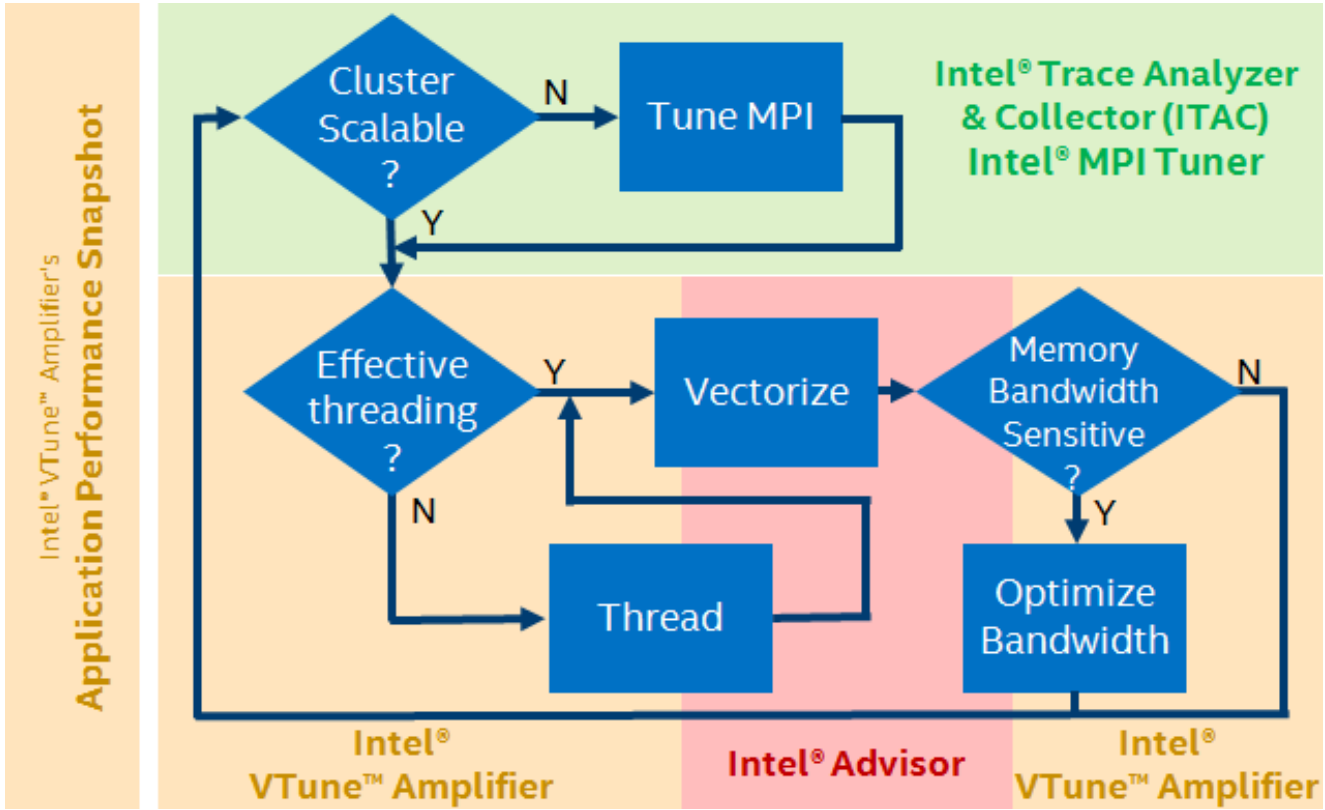


Figure: Intel Methodology to achieve performance

Intel Advisor tool

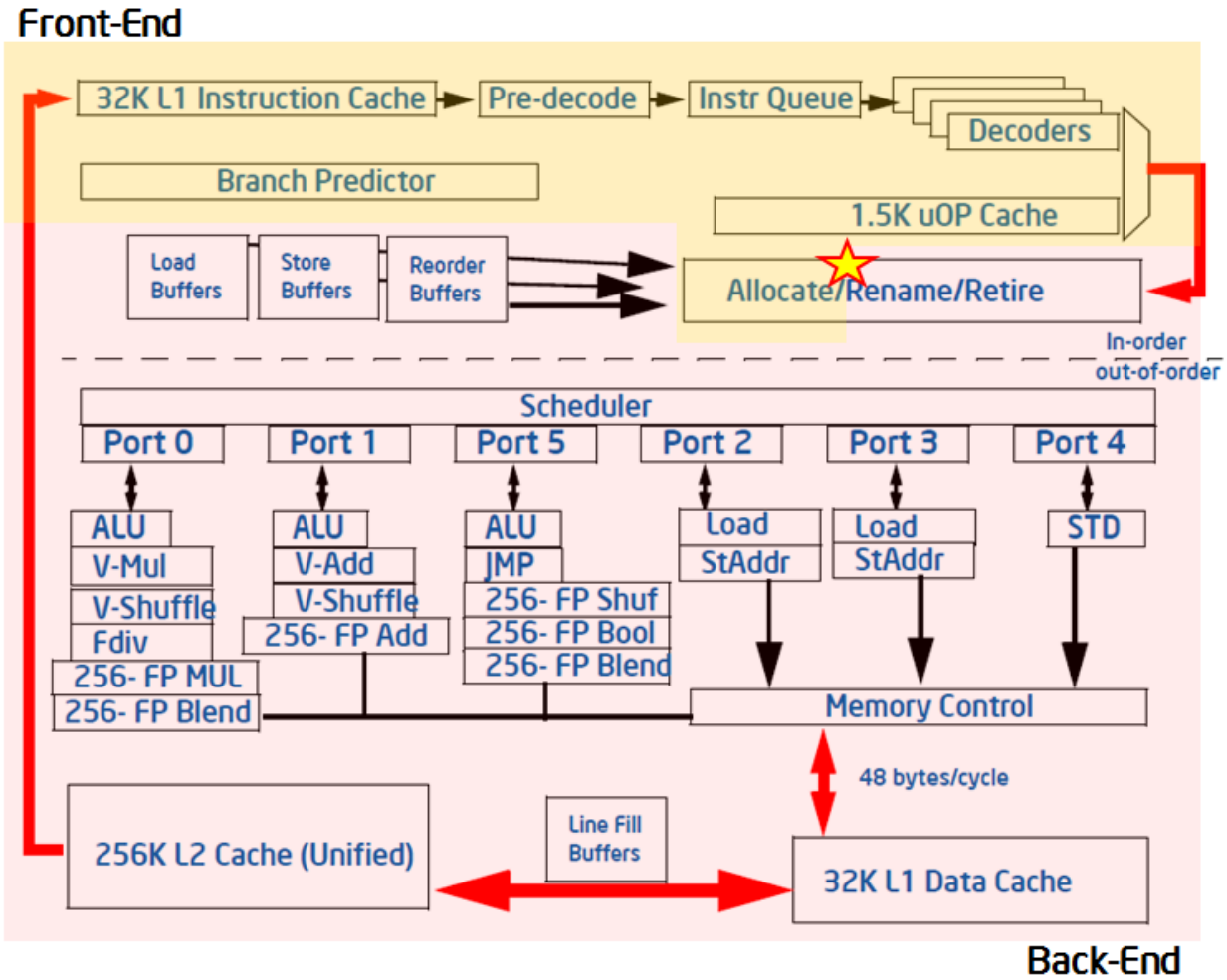


Figure: Intel Methodology to achieve performance

Intel Advisor tool

- and beyond ...

Intel Advisor tool

With MPI - do it in 2 steps:

- mpirun -np 1 advixe-cl -collect survey –project-dir=wave0 –ignore-checksums –no-auto-finalize ./bin/wave0 5 5 5 100 100 100 0.0005 500

Intel Advisor tool

With MPI - do it in 2 steps:

- mpirun -np 1 advixe-cl -collect survey –project-dir=wave0 –ignore-checksums –no-auto-finalize ./bin/wave0 5 5 5 100 100 100 0.0005 500
- mpirun -np 1 advixe-cl –collect tripcounts –ignore-checksums –project-dir=wave0 –flop –no-trip-counts – ./bin/wave0 5 5 5 100 100 100 0.0005 500

Intel Advisor tool

With MPI - do it in 2 steps:

- mpirun -np 1 advixe-cl -collect survey –project-dir=wave0 –ignore-checksums –no-auto-finalize ./bin/wave0 5 5 5 100 100 100 0.0005 500
- mpirun -np 1 advixe-cl –collect tripcounts –ignore-checksums –project-dir=wave0 –flop –no-trip-counts – ./bin/wave0 5 5 5 100 100 100 0.0005 500
- one trace per rank

Intel Advisor tool

- try it with hou10ni ☺

Intel Advisor tool

- guix environment –pure maphys –ad-hoc maphys pastix starpu vim – /bin/bash –norc
- export PATH=$PATH:/cm/shared/modules/intel/ivybridge/parallel_studio/2019_update4/advisor/bin64
- mpirun -np 1 advixe-cl -collect survey –project-dir=Hou10ni –ignore-checksums –no-auto-finalize ./hou10ni_lite.out ¡ param_simple_maphys.txt
- mpirun -np 1 advixe-cl –collect tripcounts –ignore-checksums –project-dir=Hou10ni –flop –no-trip-counts – ./hou10ni_lite.out ¡ param_simple_maphys.txt

Intel Advisor tool

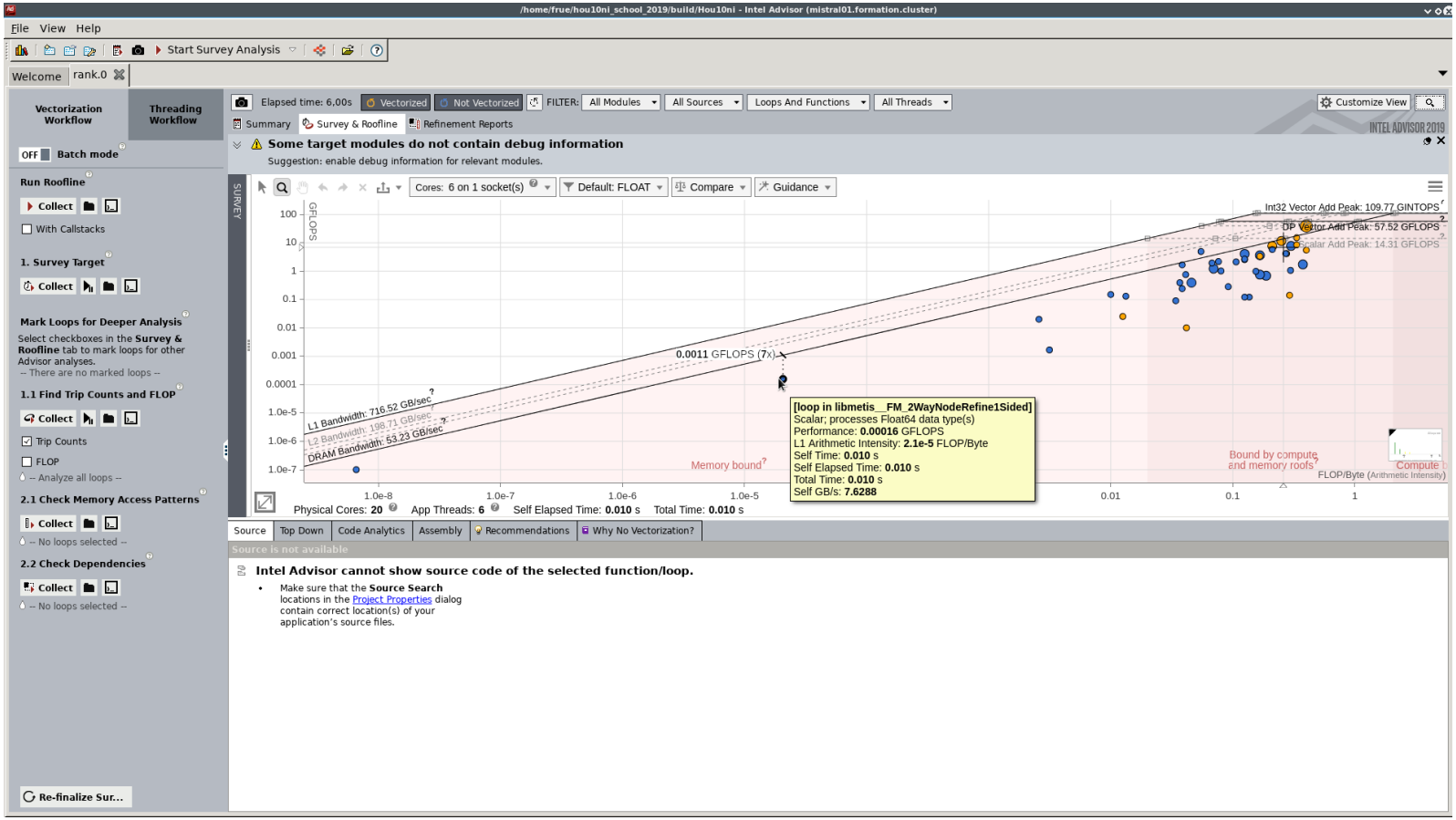


Figure: hou10ni - profiling

Intel Advisor tool

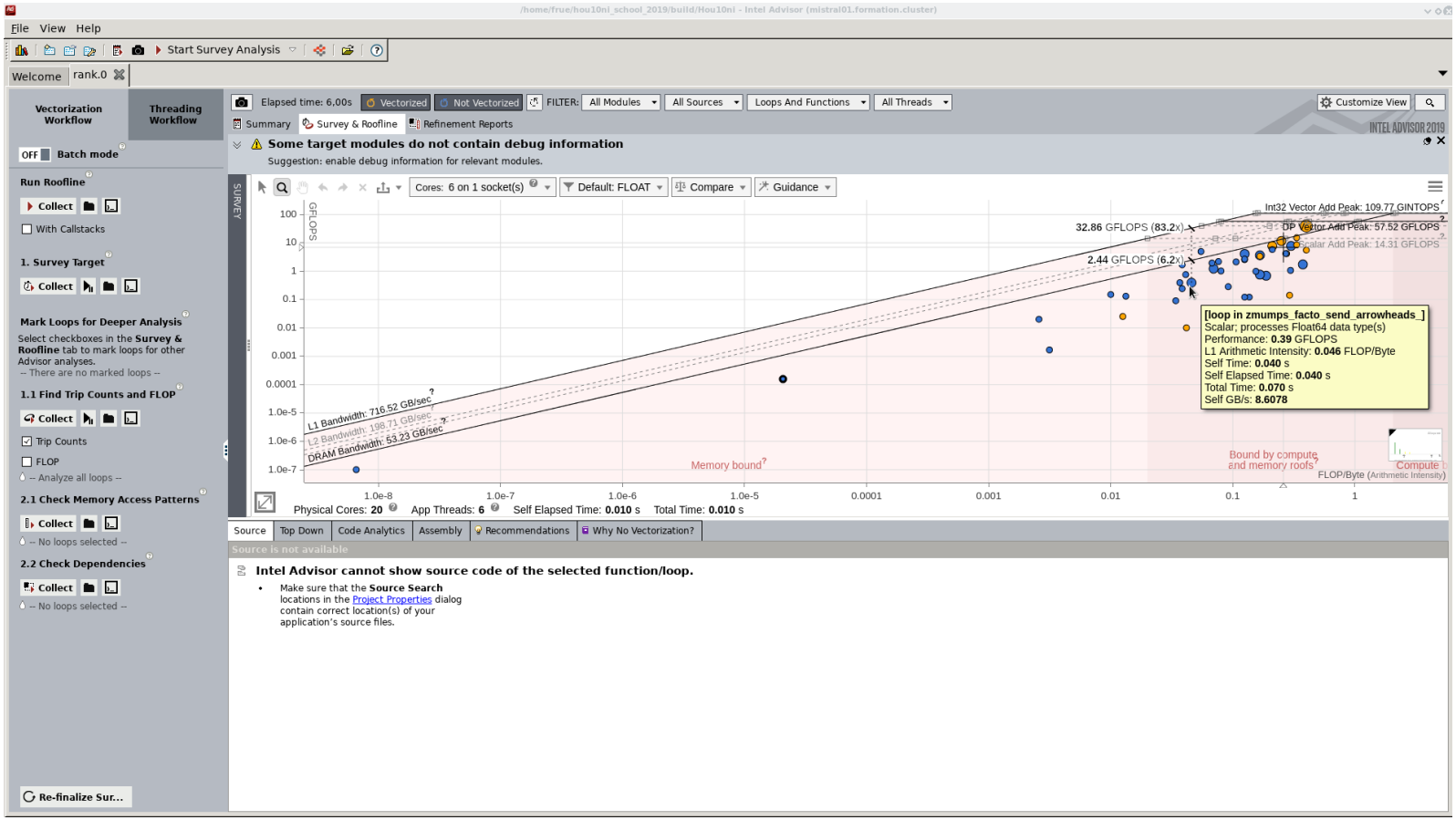


Figure: hou10ni - profiling

Intel Advisor tool

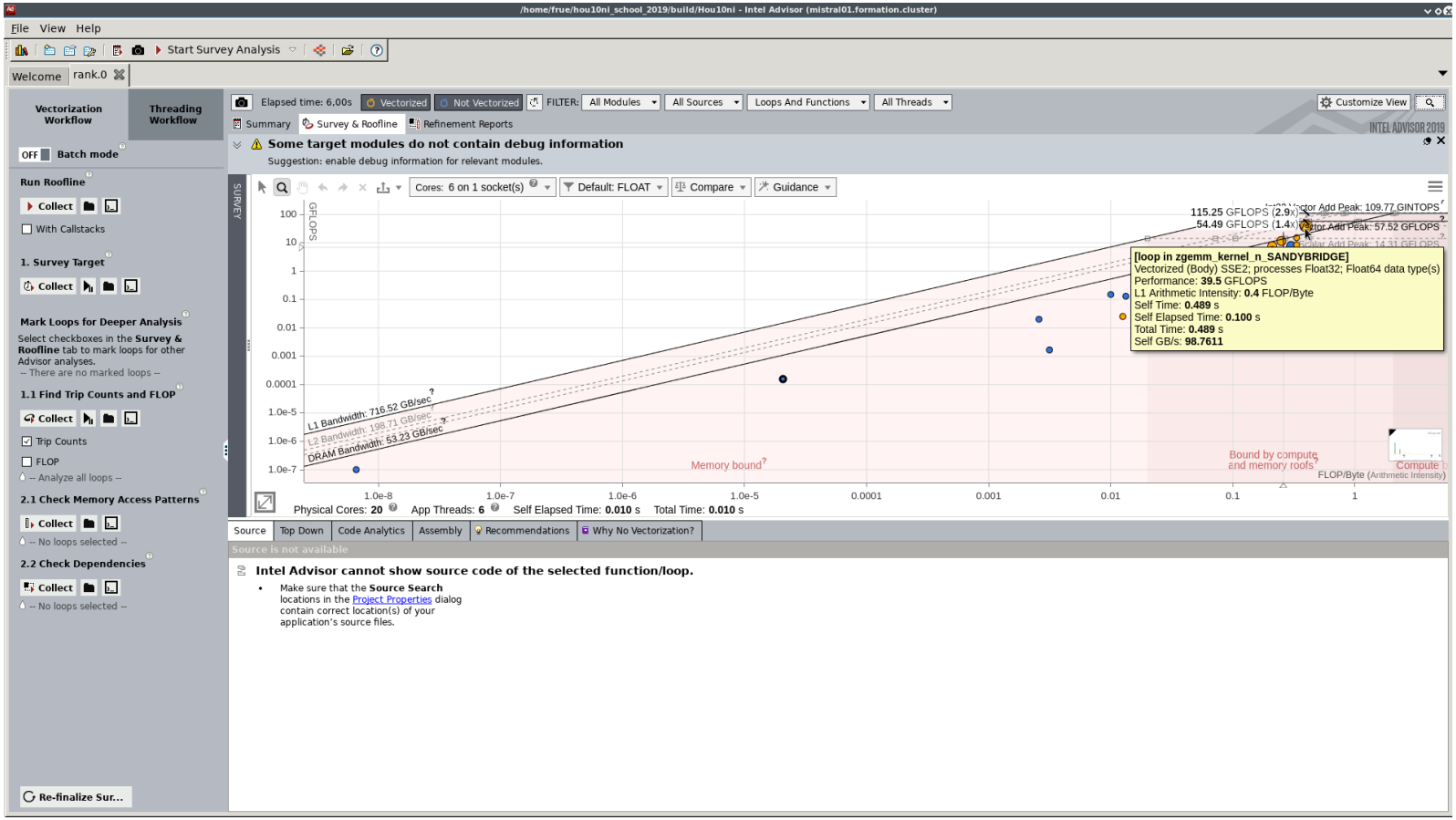


Figure: hou10ni - profiling

KEEP CALM

KEEP CALM

this is my

LAST SLIDE