

# PlaFRIM

Courtès L., Rué F.

November 8, 2019

# Table of contents

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- 1 Introduction
- 2 General Exploration
- 3 The Roofline model
- 4 Performance Methodology

# The hard way

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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

# The hard way

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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



# The optimization objectives

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- Improve the speed of execution

# The optimization objectives

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- Improve the speed of execution
- Reduce memory footprint

# The optimization objectives

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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

# The optimization objectives

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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



# The Process

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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

# How profilers do it

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- Call stack sampling

# How profilers do it

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- Call stack sampling
- Optional function call instrumentation

# How profilers do it

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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

# How profilers do it

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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

# Memory

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

## Understanding memory locality

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

# General Exploration

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

## Optimization and granularity

# The easiest way

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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



# The easiest way

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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

## static instrumentation - gprof

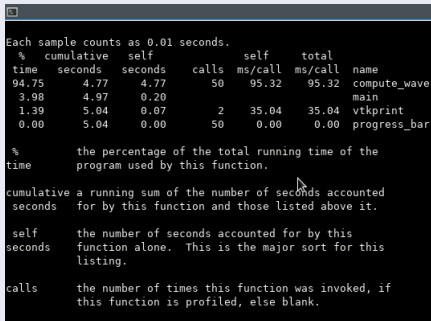
- Sampling technique
- no instrumentation needed
- 2 types of view (flat profile and call graph)

## static instrumentation - gprof

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

## 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         35.04   35.04 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

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

## 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
[1]  100.0    0.20   4.84         <spontaneous>
      4.77    0.00   50/50      main [1]
      0.07    0.00    2/2      compute_wave [2]
      0.00    0.00   50/50      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

PlaFRIM

Courtès L.,  
Rué F.

Introduction

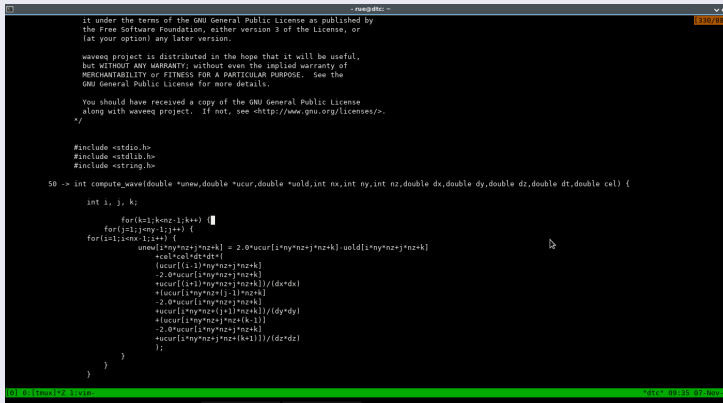
General  
Exploration

The Roofline  
model

Performance  
Methodology

## static instrumentation - gprof

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



```
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

waveeq project is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with waveeq project. If not, see <http://www.gnu.org/licenses/>.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

50 -> int compute_wave(double *unew,double *ucur,double *uold,int nx,int ny,int nz,double dx,double dy,double dz,double dt,double cel) {

    int i, j, k;

    for(k=1;k<nx-1;k++) {
        for(j=1;j<ny-1;j++) {
            for(i=1;i<nx-1;i++) {
                unew[i*ny*nz+j*nz+k] = 2.0*ucur[i*ny*nz+j*nz+k]-uold[i*ny*nz+j*nz+k]
                +cel*(cel*dt*dt*(
                    (ucur[(i-1)*ny*nz+j*nz+k]
                    -2.0*ucur[i*ny*nz+j*nz+k]
                    +ucur[(i+1)*ny*nz+j*nz+k])/(dx*dx)
                    +(ucur[i*ny*nz+(j-1)*nz+k]
                    -2.0*ucur[i*ny*nz+j*nz+k]
                    +ucur[i*ny*nz+(j+1)*nz+k])/(dy*dy)
                    +(ucur[i*ny*nz+j*nz+(k-1)]
                    -2.0*ucur[i*ny*nz+j*nz+k]
                    +ucur[i*ny*nz+j*nz+(k+1)])/(dz*dz)
                ));
            }
        }
    }

    (C) 0.17maxi+2 livin... *dtc* 00:35 07-Nov-2013
```

# Profiler

PlaFRIM

Courtès L.,  
Rué F.

Introduction

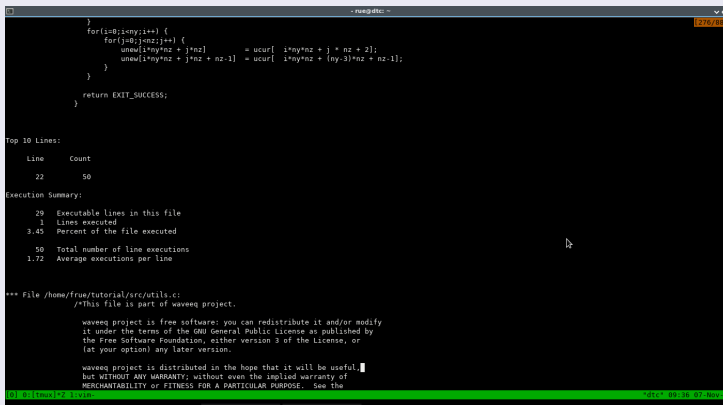
General  
Exploration

The Roofline  
model

Performance  
Methodology

## static instrumentation - gprof

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



```

}
for(i=0; i<ny; i++) {
    for(j=0; j<nz; j++) {
        unew[i*ny*nz + j*nz] = ucw[ i*ny*nz + j * nz + 2];
        unew[i*ny*nz + j*nz + nz-1] = ucw[ i*ny*nz + (ny-3)*nz + nz-1];
    }
}
return EXIT_SUCCESS;
}

Top 10 Lines:

    Line    Count
    ----    -
    22         50

Execution Summary:

    29 Executable lines in this file
     1 Lines executed
    3.45 Percent of the file executed
    50 Total number of line executions
    1.72 Average executions per line

*** File /home/frue/tutorial/src/utls.c:
/*This file is part of waveeq project.

waveeq project is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

waveeq project is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

```

# Profiler

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- and for memory usage ?



## Dynamic instrumentation - valgrind

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

## Profiler

PlaFRIM

Courtès L.,  
Rué F.

Introduction

**General  
Exploration**

The Roofline  
model

## General Exploration

## 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`

[illegible]

# Profiler

## PIaFRIM

## General Exploration

## 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`

[illegible]

# Profiler

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

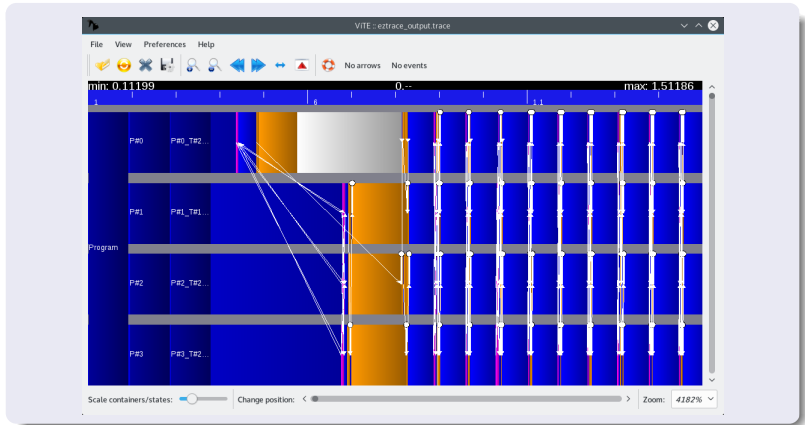
- what kind of expertise ?

# Profiler

PlaFRIM

Courtès L.,  
Rué F.

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



# Profiler

PlaFRIM

Courtès L.,  
Rué F.

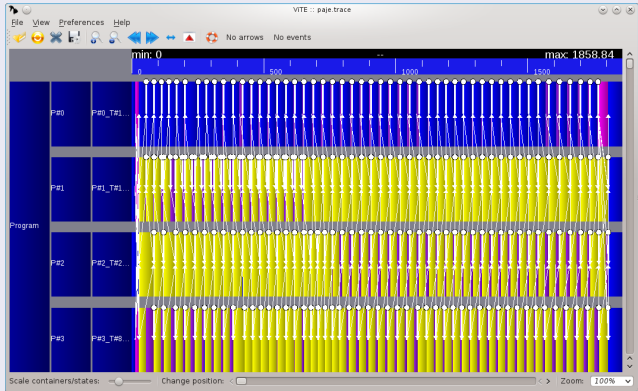
Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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



# The Roofline model

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

**The Roofline  
model**

Performance  
Methodology

roofline

# The model

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

## cache aware roofline model

Performance Limiting Factors

IBM

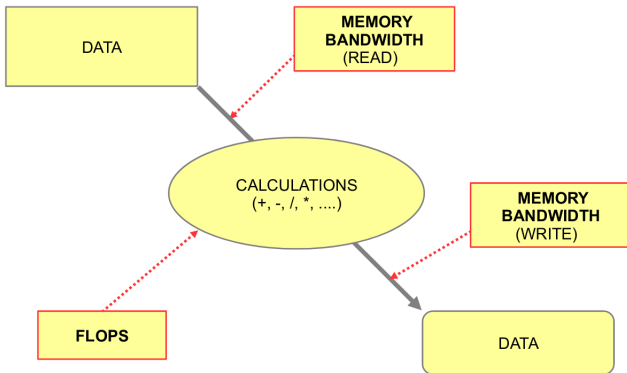


Figure: IBM - ICSC 2014, Shanghai, China



# The model

PlaFRIM

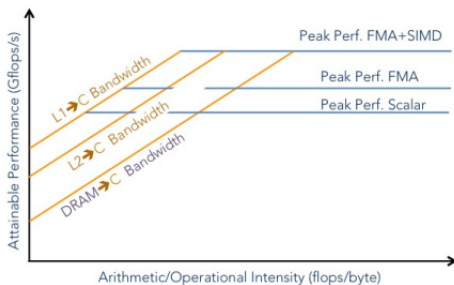
Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology



•Total volume communication across all memory hierarchies relatively to the core:

$$\text{Attainable perf Gflops/s} = \min \left\{ \begin{array}{l} \text{Peak performance Gflops/s} \\ \text{Bandwidths to Core} \times \text{Arithmetic Intensity} \end{array} \right.$$

$$\text{Bandwidths to Core} = \left\{ \begin{array}{l} \text{Bandwidth from L1 to Core} \\ \text{Bandwidth from L2 to Core} \\ \text{Bandwidth from L3 to Core} \\ \text{Bandwidth from DRAM to Core} \end{array} \right.$$

Figure: PICSAR Project

# The model

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- AXPY:  $y[i] = a * x[i] + y[i]$ ,  $a$  is real,  $i=0 \dots N-1$
- 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':  $2 \times 4 = 8$  bytes
    - keep 'a' in a register
  - need to write out  $y[i]$ : another 4 bytes
  - Arithmetic Intensity:  $2 \text{ FLOPS} / 12 \text{ bytes} = 1/6$
  - 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

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

## cache aware roofline model

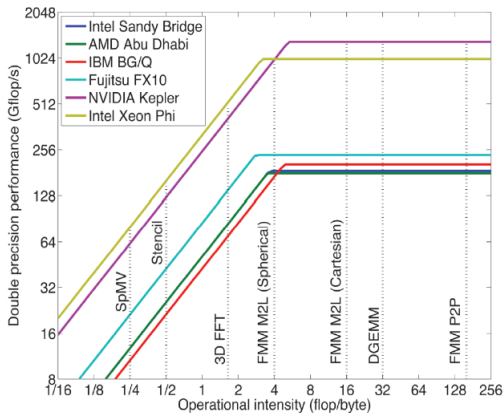


Figure: IBM - ICSC 2014, Shanghai, China

# The model

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- How to construct this model ?

# The model

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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

# Roofline evaluation

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

...

# Roofline evaluation

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

evaluate the performance you can achieve

## Performance achievement

## Understanding memory locality

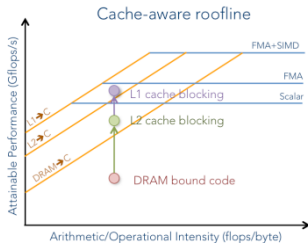


Figure: Memory Bound

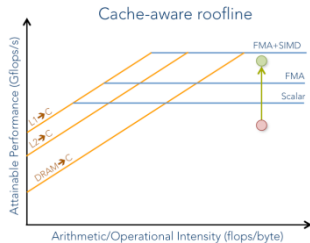


Figure: Compute Bound



# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

one tool to do that ...

# Intel Advisor tool

PlaFRIM

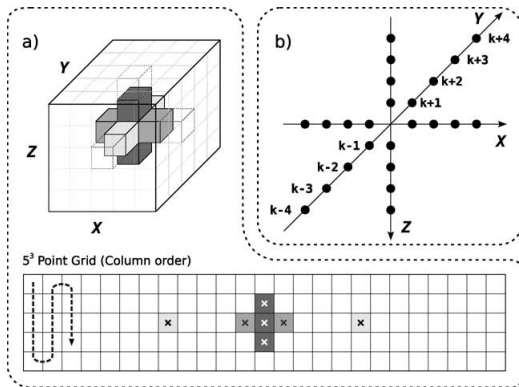
Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology



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

## Intel Advisor tool

PlaFRIM

Performance Methodology

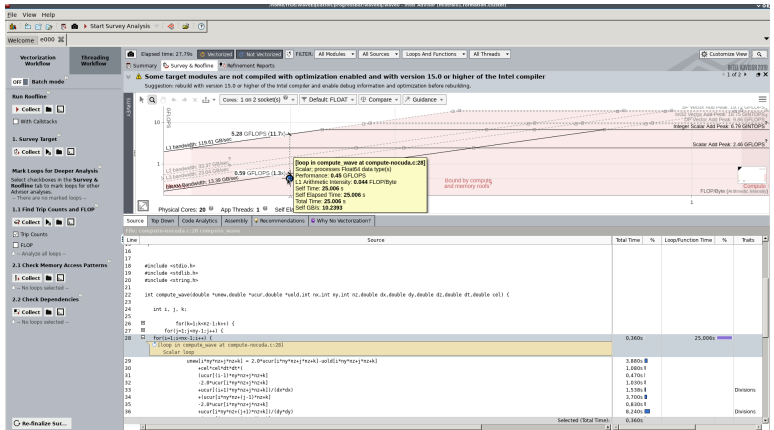


Figure: Stencil 1 thread - roofline

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- `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

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- `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

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

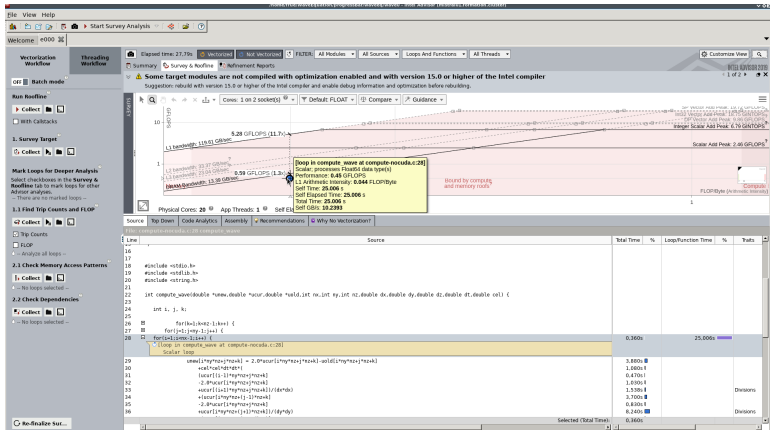


Figure: Stencil 1 thread - roofline

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

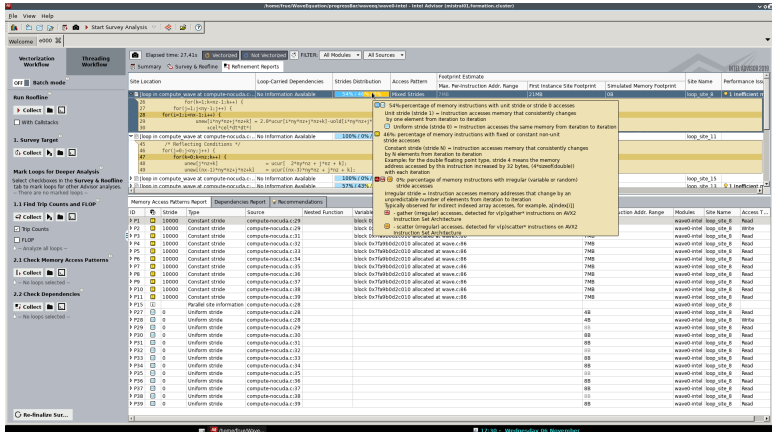


Figure: Stencil 1 thread - memory access pattern

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

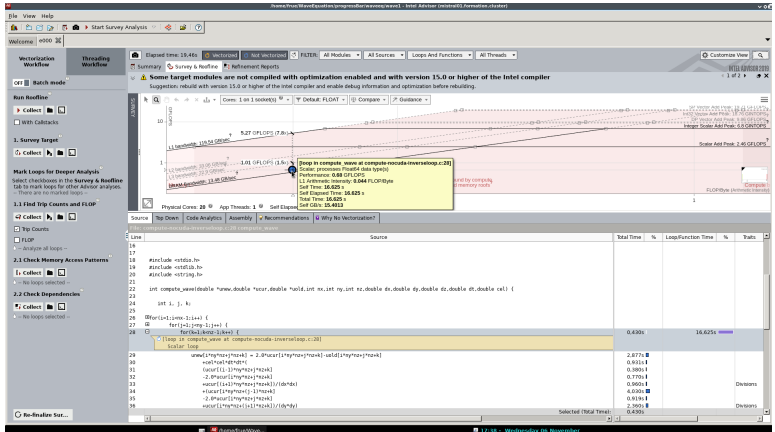


Figure: Stencil 1 thread - inverse loop - roofline



# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- strides distribution - better performance

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- strides distribution - better performance
- cache blocking technic?

# Intel Advisor tool

## PIaFRIM

## Introduction

## Performance Methodology

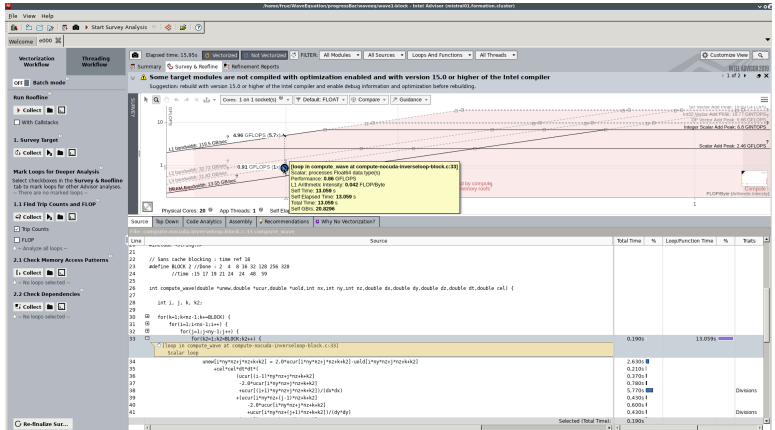


Figure: Stencil 1 thread - inverse loop & cache blocking - roofline

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- OpenMP ?

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

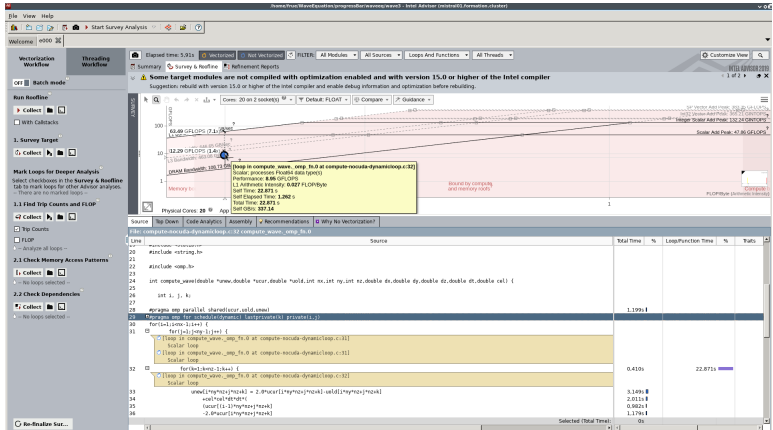


Figure: Stencil 20 threads - inverse loop & OpenMP - roofline

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

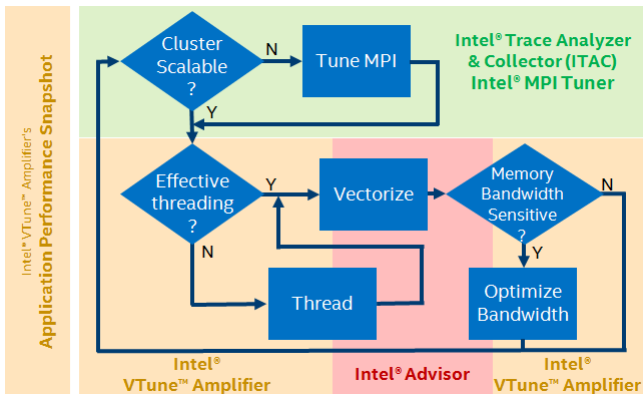


Figure: Intel Methodology to achieve performance

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

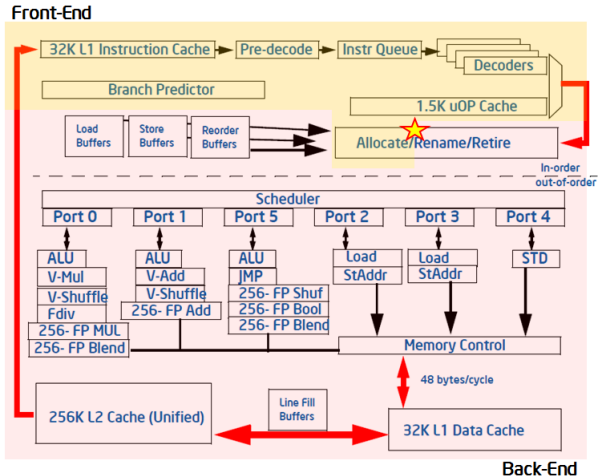


Figure: Intel Methodology to achieve performance

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- and beyond ...



# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

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

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- try it with hou10ni 😊

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

- `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 j param_simple_maphys.txt`
- `mpirun -np 1 advixe-cl --collect tripcounts --ignore-checksums --project-dir=Hou10ni --flop`  
`--no-trip-counts -- ./hou10ni_lite.out j param_simple_maphys.txt`

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

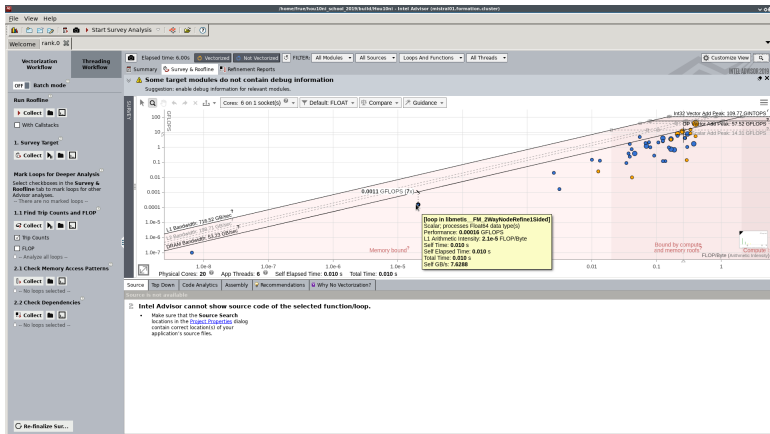


Figure: hou10ni - profiling

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

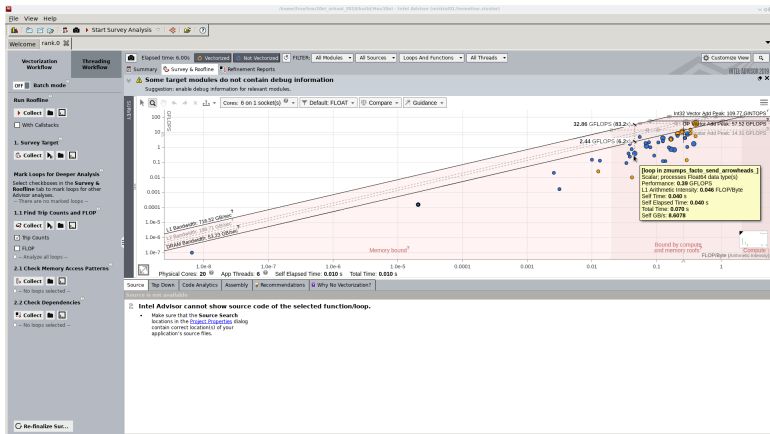


Figure: hou10ni - profiling

# Intel Advisor tool

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

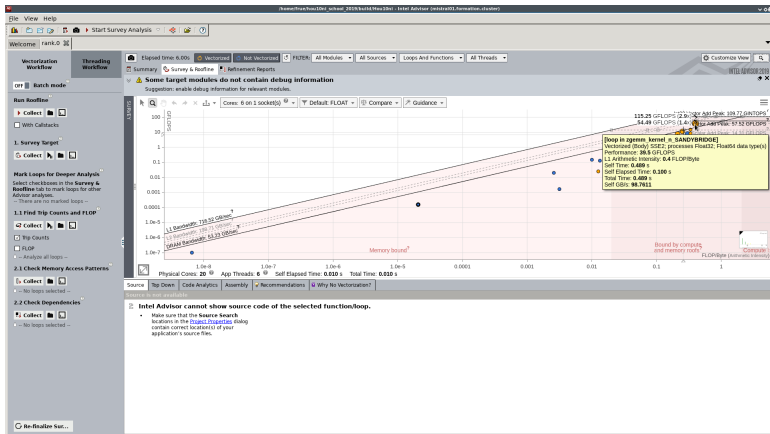


Figure: hou10ni - profiling



PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

KEEP CALM

PlaFRIM

Courtès L.,  
Rué F.

Introduction

General  
Exploration

The Roofline  
model

Performance  
Methodology

KEEP CALM

this is my

LAST SLIDE