



Intel performance analysis tools for HPC

Mathieu Lobet, Matthieu Haefele

Maison de la Simulation, CEA, CNRS, Université Paris-Sud, UVSQ,
Universite Paris-Saclay, F-91191 Gif-sur-Yvette, France
(mathieu.lobet@cea.fr)



Intel Advisor



Advisor description

Intel Advisor : Vectorization advisor and Threading advisor

Vectorization advisor is a vectorization optimization tool:

Help to identify time-consuming loops that can benefit from vectorization or already vectorized

Help to identify vectorization and efficiency issues (dependencies, spilling, memory access...) and propose solutions

Help to ensure that vectorization is safe and quantify effects of vectorization (vectorization efficiency, roofline performance model)



Compilation for Advisor

Classical compilation with additional flags:

- -g
- -debug inline-debug-info
- -shared-intel
- -parallel-source-info=2

<https://software.intel.com/en-us/vtune-amplifier-help-compiler-switches-for-performance-analysis-on-linux-targets>



Execution with Advisor

```
> mpirun -np 1 advixe-cl -collect $ANALYSIS --project-dir  
$ANALYSIS_DIR --search-dir src:r=$SRCDIR -verbose -flop - ./  
$APPNAME
```



Execution with Advisor

```
> mpirun -np 1 advixe-cl -collect $ANALYSIS --project-dir  
$ANALYSIS_DIR --search-dir src:r=$SRCDIR -verbose -flop - ./  
$APPNAME
```

\$ANALYSIS=survey: general overview of the performances and the vectorization state of the code.



Execution with Advisor

```
> mpirun -np 1 advixe-cl -collect $ANALYSIS --project-dir  
$ANALYSIS_DIR --search-dir src:r=$SRCDIR -verbose -flop - ./  
$APPNAME
```

\$ANALYSIS=tripcounts: improves the survey by dynamically exploring loop iteration execution and propose better decisions about your vectorization strategy. It measures #FLOP count and cumulative data traffic necessary for the Roofline performance model.



Execution with Advisor

```
> mpirun -np 1 advixe-cl -collect $ANALYSIS --project-dir  
$ANALYSIS_DIR --search-dir src:r=$SRCDIR -verbose -flop - ./  
$APPNAME
```

\$ANALYSIS=dependencies: refine analysis by checking for real data dependencies in loops the compiler did not vectorize because of assumed dependencies.



Execution with Advisor

```
> mpirun -np 1 advixe-cl -collect $ANALYSIS --project-dir  
$ANALYSIS_DIR --search-dir src:r=$SRCDIR -verbose -flop - ./  
$APPNAME
```

\$ANALYSIS=map: (Memory Access Pattern) refine analysis by checking for various memory issues, such as non-contiguous memory accesses and unit stride vs. non-unit stride accesses.



Execution with Advisor

```
> mpirun -np 1 advixe-cl -collect $ANALYSIS --project-dir  
$ANALYSIS_DIR --search-dir src:r=$SRCDIR -verbose -flop - ./  
$APPNAME
```

-no-auto-finalize: Results are not finalized after the run. This is useful for collections on KNL that takes longer time than on Haswell. Finalization can then be performed on Haswell or when opening the results via the GUI.



Execution with Advisor

```
> mpirun -np 1 advixe-cl -collect $ANALYSIS --project-dir  
$ANALYSIS_DIR --search-dir src:r=$SRCDIR -verbose -flop - ./  
$APPNAME
```

-flop: take into account the masks in the vector operations on AVX512



Execution with Advisor

```
> mpirun -np 1 advixe-cl -collect $ANALYSIS --project-dir  
$ANALYSIS_DIR --search-dir src:r=$SRCDIR -verbose -flop - ./  
$APPNAME
```

: enable to target specific loops for the analysis



Visualization of the results

```
> advixe-gui ./$ANALYSIS_DIR
```

Open the graphic interface to visualilze the results



Visualization of the results

Global interface

File View Help

Start Survey Analysis

Welcome rank.0

Vectorization Workflow Threading Workflow

OFF Batch mode

Run Roofline

Collect

Enable Roofline with Callstacks

1. Survey Target

Collect

Mark Loops for Deeper Analysis

Select checkboxes in the Survey & Roofline tab to mark loops for other Advisor analyses.

-- There are no marked loops --

1.1 Find Trip Counts and FLOP

Collect

Trips Counts

FLOP

-- Analyze all loops --

2.1 Check Memory Access Patterns

Collect

-- No loops selected --

2.2 Check Dependencies

Collect

-- No loops selected --

Re-finalize Sur...

Elapsed time: 0,93s

Vectorized Not Vectorized

FILTER: All Modules All Sources

Summary Survey & Roofline Refinement Reports

Vectorization Advisor

Vectorization Advisor is a vectorization analysis toolset that lets you identify loops that will benefit most from vector parallelism, discover performance issues preventing from effective vectorization and characterize your memory vs. vectorization bottlenecks with Advisor Roofline model automation.

Program metrics

Elapsed Time 0,93s

Vector Instruction Set AVX

Number of CPU Threads 1

Total GFLOP Count 0,05

Total GFLOPS 0,05

Total Arithmetic Intensity 0,18230

Loop metrics

Metrics	Total
Total CPU time	0,16s 100,0%
Time in 3 vectorized loops	0,07s 43,7%
Time in scalar code	0,09s 56,3%
Total GFLOP Count	0,05 100,0%
Total GFLOPS	0,05

Vectorization Gain/Efficiency

Vectorized Loops Gain/Efficiency 3,76x 94%

Program Approximate Gain 2,21x

Top time-consuming loops

Loop	Self Time	Total Time	Trip Counts
[loop in move_... at parti_...]	0,050s	0,050s	22589; 3
[loop in init_... at parti_...]	0,010s	0,010s	524288
[loop in depos_... at charg_...]	0,010s	0,010s	90359
[loop in init_... at parti_...]	0,010s	0,010s	65536
[loop in main at main_...]	0s	0,060s	10

Refinement analysis data

These loops were analyzed for memory access patterns and dependencies:

Site Location	Dependencies	Strides Distribution
[loop in depos_... at charg_...]	RAW:1	0% / 27% / 73%
[loop in move_... at parti_...]	RAW:1	71% / 10% / 19%

Collection details

Platform information

MPI rank 0



Visualization of the results

Survey interface

Elapsed time: 0,93s Vectorized Not Vectorized FILTER: All Modules All Sources Loops All Threads OFF Smart Mode

Summary Survey & Roofline Refinement Reports

Function Call Sites and Loops	Performance Issues	Self Time	Total Time	Type	Why No Vectorization?	Vectorized Loops				Com...	Self AI	Self Memory...	Trip Counts		Instruct
						Vect...	Efficiency	Gain...	VL (...)				Average	Call Count	
[loop in move_particles at particles.F90:93]	2 Possible i...	0,050s	0,050s	Vectorized (B...		AVX	91%	3,62x	4	0,506	0,24561	0,103	22589; 3	10; 10	Division
[loop in move_particles at particles.F90:93]	1 Possible ine...	0,050s	0,050s	Vectorized (Body)		AVX			4	0,506	0,24561	0,103	22589	10	Division
[loop in move_particles at particles.F90:93]	1 Unoptimize...	0,000s	0,000s	Remainder							0,26923	< 0,001	3	10	Division
[loop in initialize_particles at particles.F90:35]		0,010s	0,010s	Vectorized (Body)		AVX	100%	5,17x	4			0,017	524288	1	
[loop in deposit_charge at charge.F90:31]	2 Assumed d...	0,010s	0,010s	Scalar	vector dependence ...					1,886	0,21591	0,087	90359	11	Division
[loop in initialize_particles at particles.F90:38]		0,010s	0,010s	Vectorized (Body)		AVX	77%	3,06x	4	0,421	0,12500	0,034	65536	1	Extract
[loop in rxm_endpoint]	1 System fun...	0,000s	0,012s	Scalar								< 0,001	7	1	
[loop in fi_ini]	2 System fun...	0,000s	0,030s	Scalar								< 0,001	4	1	
[loop in fi_ini]		0,000s	0,030s	Scalar								< 0,001	1	1	
[loop in off_cq_progress]	1 Indirect fu...	0,000s	0,010s	Scalar								< 0,001	1	36625	
[loop in main at main.F90:90]		0,000s	0,060s	Scalar	loop control variabl...							< 0,001	10	1	
[loop in move_particles at particles.F90:97]		n/a	n/a	Inside vectorize...											
[loop in move_particles at particles.F90:101]		n/a	n/a	Inside vectorize...											
[loop in move_particles at particles.F90:101]		n/a	n/a	Inside vectorize...											
[loop in deposit_charge at charge.F90:34]		n/a	n/a	Scalar Complet...											

Source Top Down Code Analytics Assembly Recommendations Why No Vectorization?

File: particles.F90:93 move_particles

Line	Source
85	! Local variable to store the electric field
86	real(8), dimension(2) :: E
87	! Additional parameters for the field
88	real(8) :: r,sin,cos
89	
90	! Constant that will be used by each particle
91	cst = species%charge*subdomain%delta_time / species%mass
92	
93	do i = 1,subdomain%number_of_particles
	[loop in move_particles at particles.F90:93]
	Vectorized AVX loop processes Float32; Float64; UInt64 data type(s) and includes Divisions; Inserts; Permutes; Square Roots; Unpacks
	No loop transformations applied
	[loop in move_particles at particles.F90:93]
	Scalar remainder loop with instructions that use AVX registers
	No loop transformations applied
94	
95	! Field computation
96	

Selected (Total Time):



Useful links

- Compilation flags: <https://software.intel.com/en-us/vtune-amplifier-help-compiler-switches-for-performance-analysis-on-linux-targets>
- Command line interface: <https://software.intel.com/en-us/advisor-user-guide-intel-advisor-cli#F673BC4E-187B-4A08-AAF9-366304468295>
- Command line interface: <https://software.intel.com/en-us/advisor-user-guide-command-line-interface-reference>
- Collection description: <https://software.intel.com/en-us/advisor-user-guide-collect>